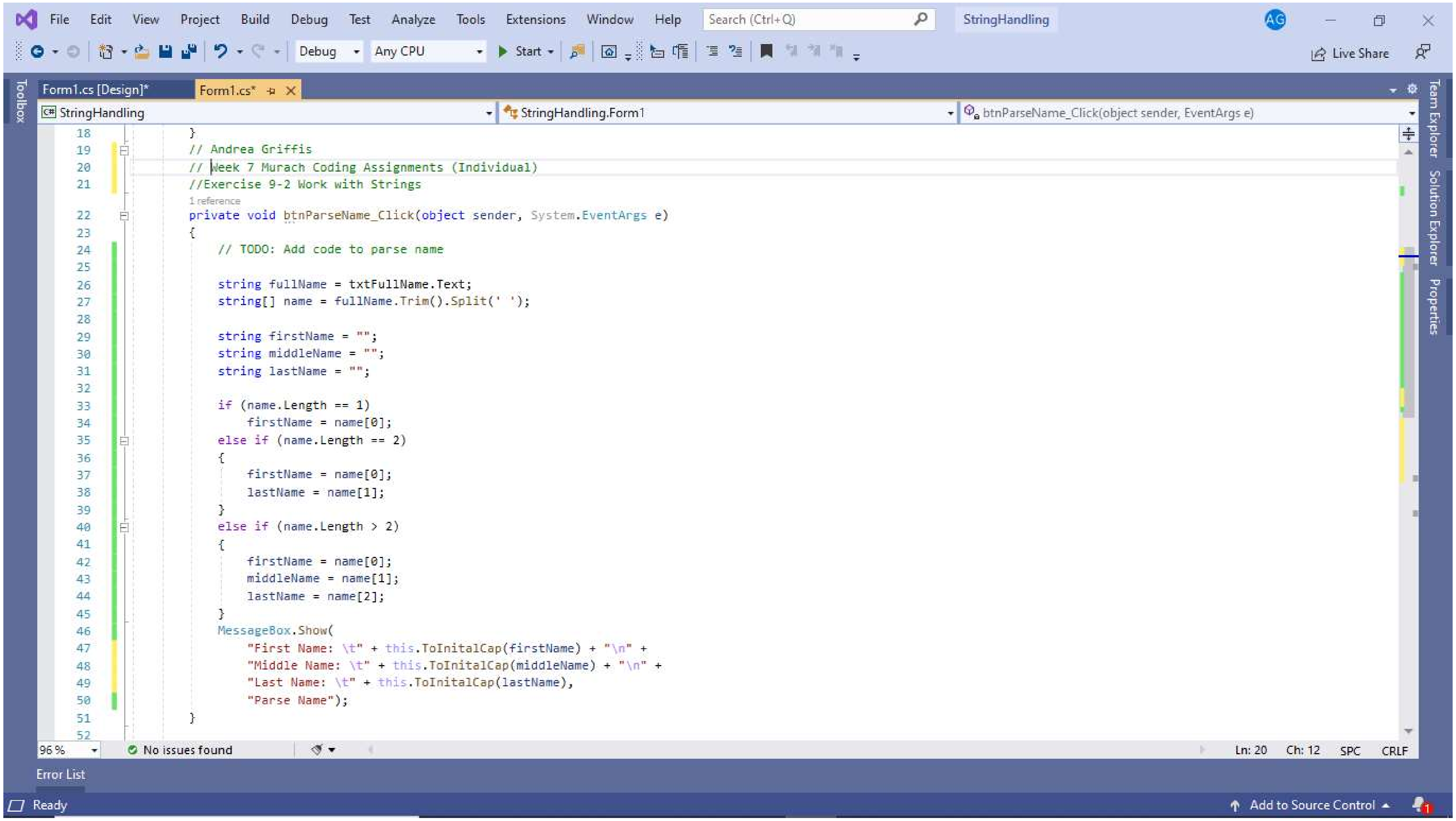
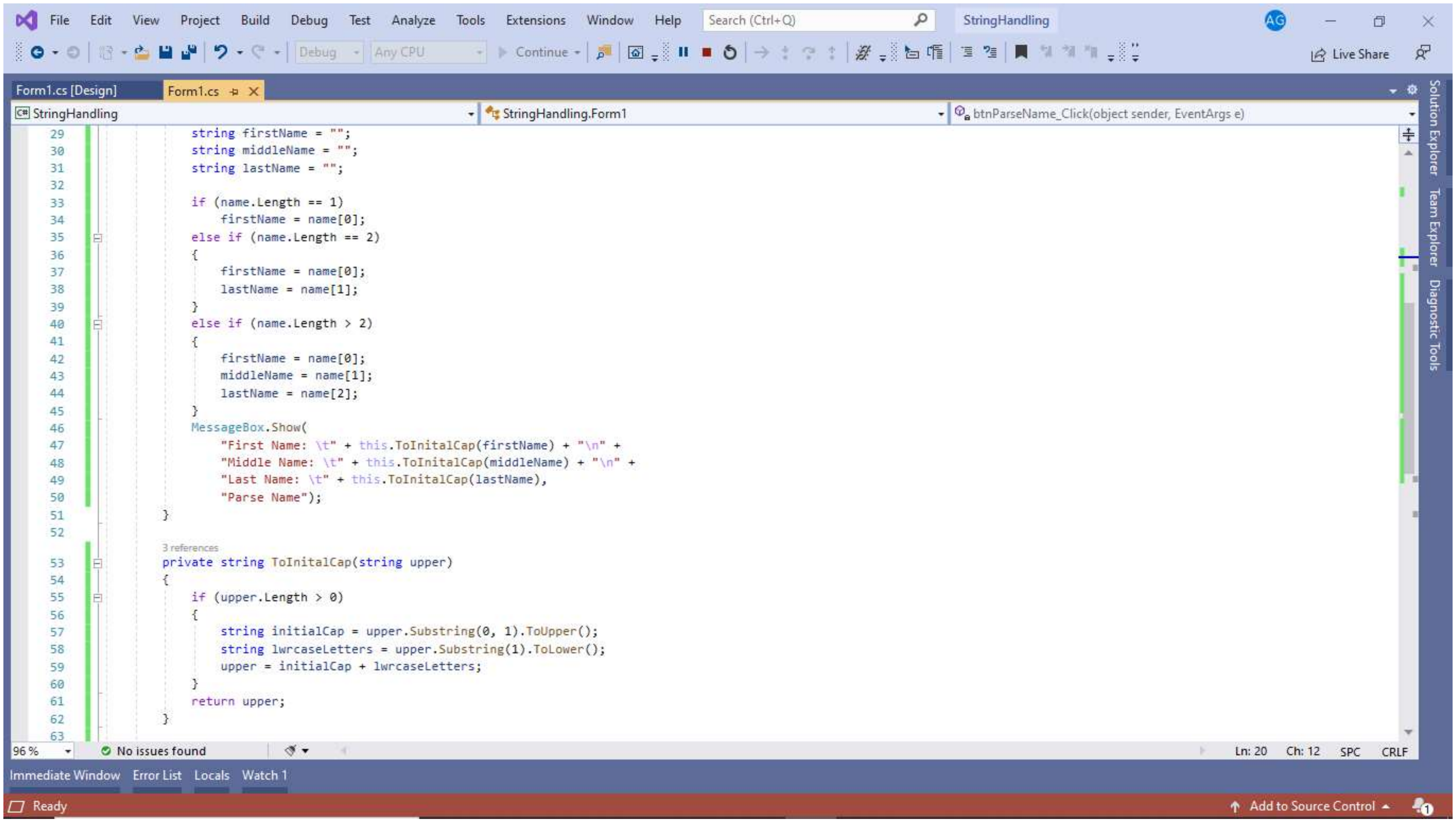
Andrea Griffis
CIS 123 C #
Assignment: Week 7 Murach Coding Assignments (Individual)

Exercise 9-2 Work with Strings

Add code to parse a name

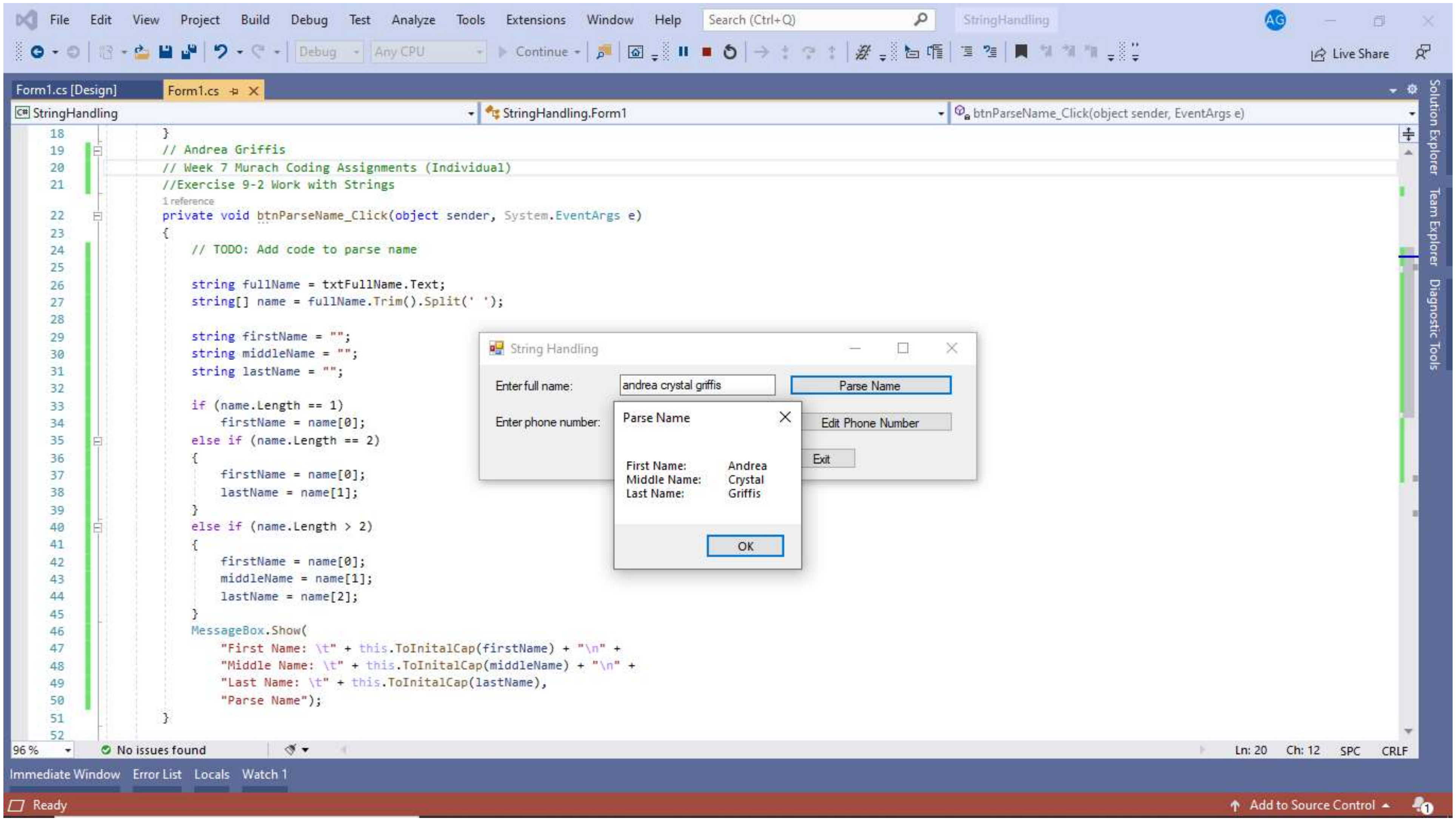


```
        }
        // Andrea Griffis
        // Week 7 Murach Coding Assignments (Individual)
        //Exercise 9-2 Work with Strings
        1 reference
        private void btnParseName_Click(object sender, System.EventArgs e)
        {
            // TODO: Add code to parse name

            string fullName = txtFullName.Text;
            string[] name = fullName.Trim().Split(' ');

            string firstName = "";
            string middleName = "";
            string lastName = "";

            if (name.Length == 1)
                firstName = name[0];
            else if (name.Length == 2)
            {
                firstName = name[0];
                lastName = name[1];
            }
            else if (name.Length > 2)
            {
                firstName = name[0];
                middleName = name[1];
                lastName = name[2];
            }
            MessageBox.Show(
                "First Name: \t" + this.ToInitalCap(firstName) + "\n" +
                "Middle Name: \t" + this.ToInitalCap(middleName) + "\n" +
                "Last Name: \t" + this.ToInitalCap(lastName),
                "Parse Name");
        }
```



```
            string firstName = "";
            string middleName = "";
            string lastName = "";

            if (name.Length == 1)
                firstName = name[0];
            else if (name.Length == 2)
            {
                firstName = name[0];
                lastName = name[1];
            }
            else if (name.Length > 2)
            {
                firstName = name[0];
                middleName = name[1];
                lastName = name[2];
            }
            MessageBox.Show(
                "First Name: \t" + this.ToInitalCap(firstName) + "\n" +
                "Middle Name: \t" + this.ToInitalCap(middleName) + "\n" +
                "Last Name: \t" + this.ToInitalCap(lastName),
                "Parse Name");
        }

        3 references
        private string ToInitalCap(string upper)
        {
            if (upper.Length > 0)
            {
                string initialCap = upper.Substring(0, 1).ToUpper();
                string lwrcaseLetters = upper.Substring(1).ToLower();
                upper = initialCap + lwrcaseLetters;
            }
            return upper;
        }
```

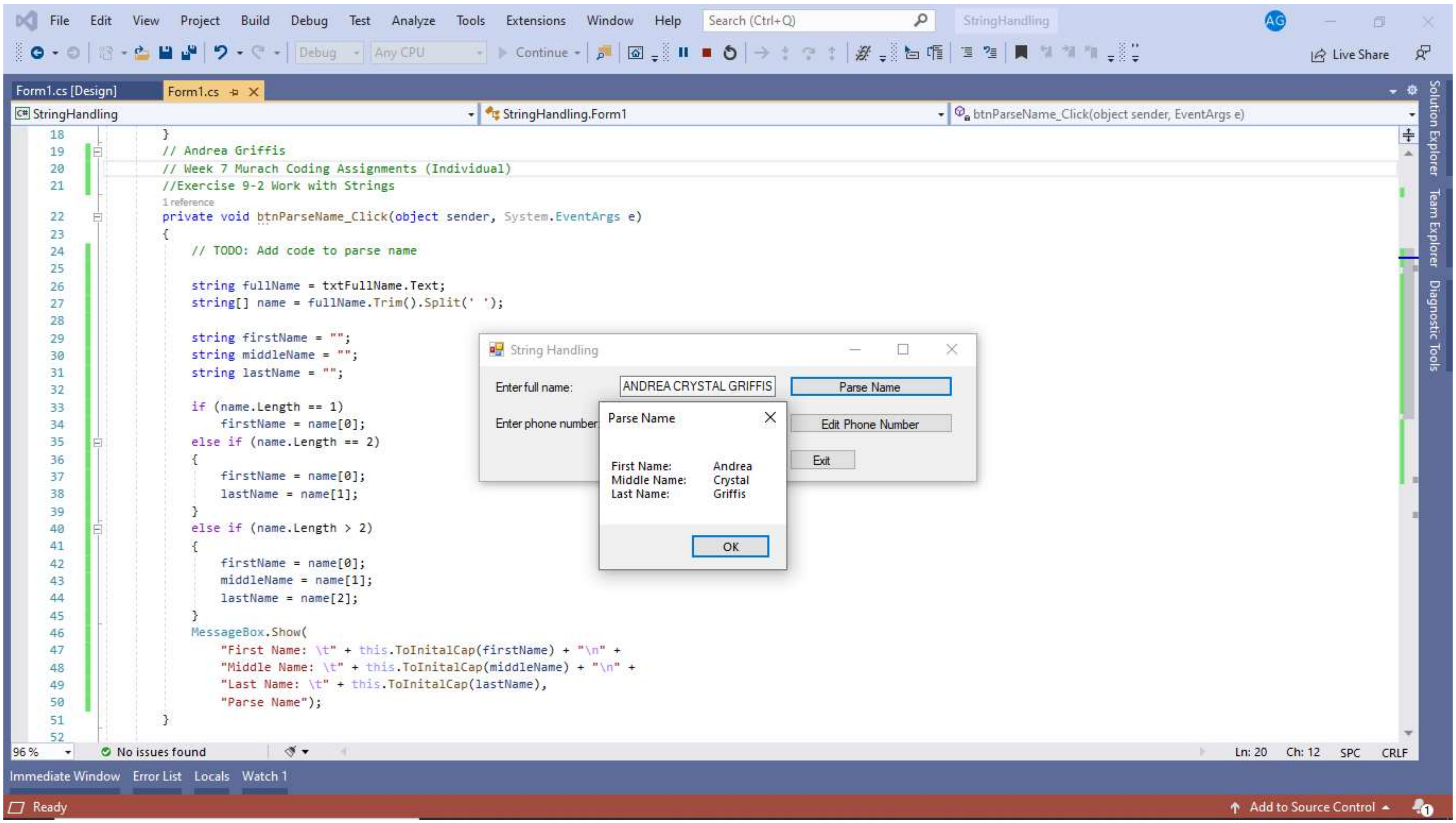
Test the Application



```
        }
        // Andrea Griffis
        // Week 7 Murach Coding Assignments (Individual)
        //Exercise 9-2 Work with Strings
        1 reference
        private void btnParseName_Click(object sender, System.EventArgs e)
        {
            // TODO: Add code to parse name

            string fullName = txtFullName.Text;
            string[] name = fullName.Trim().Split(' ');

            string firstName = "";
            string middleName = "";
            string lastName = "";

            if (name.Length == 1)
                firstName = name[0];
            else if (name.Length == 2)
            {
                firstName = name[0];
                lastName = name[1];
            }
            else if (name.Length > 2)
            {
                firstName = name[0];
                middleName = name[1];
                lastName = name[2];
            }
            MessageBox.Show(
                "First Name: \t" + this.ToInitalCap(firstName) + "\n" +
                "Middle Name: \t" + this.ToInitalCap(middleName) + "\n" +
                "Last Name: \t" + this.ToInitalCap(lastName),
                "Parse Name");
        }
```

Enter full name: andrea crystal griffis

```
First Name:     Andrea
Middle Name:    Crystal
Last Name:      Griffis
```



```
}
// Andrea Griffis
// Week 7 Murach Coding Assignments (Individual)
//Exercise 9-2 Work with Strings
1 reference
private void btnParseName_Click(object sender, System.EventArgs e)
{
    // TODO: Add code to parse name

    string fullName = txtFullName.Text;
    string[] name = fullName.Trim().Split(' ');

    string firstName = "";
    string middleName = "";
    string lastName = "";

    if (name.Length == 1)
        firstName = name[0];
    else if (name.Length == 2)
    {
        firstName = name[0];
        lastName = name[1];
    }
    else if (name.Length > 2)
    {
        firstName = name[0];
        middleName = name[1];
        lastName = name[2];
    }
    MessageBox.Show(
        "First Name: \t" + this.ToInitalCap(firstName) + "\n" +
        "Middle Name: \t" + this.ToInitalCap(middleName) + "\n" +
        "Last Name: \t" + this.ToInitalCap(lastName),
        "Parse Name");
}
```

String Handling

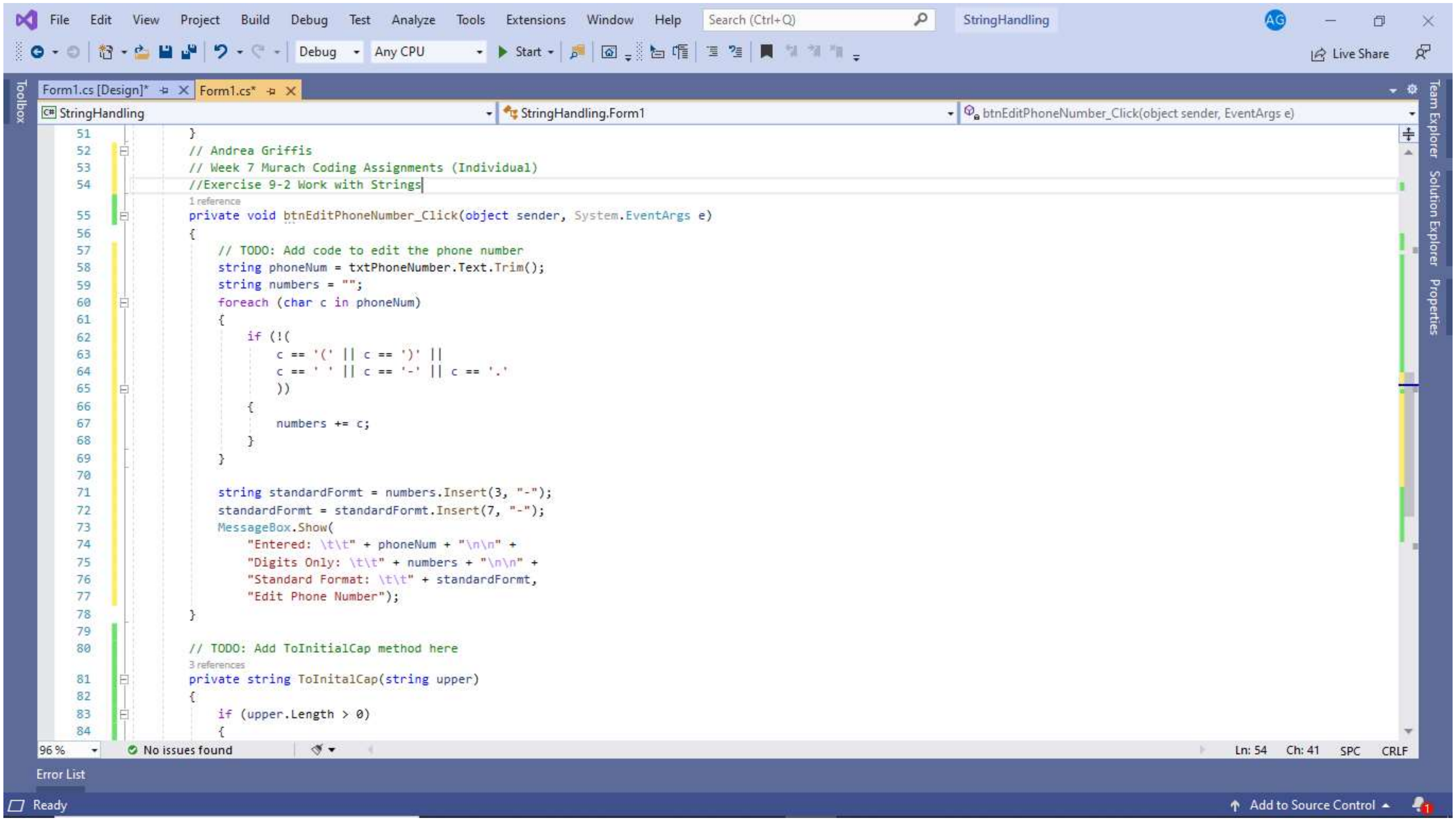
Enter full name: ANDREA CRYSTAL GRIFFIS

Enter phone number:

Parse Name

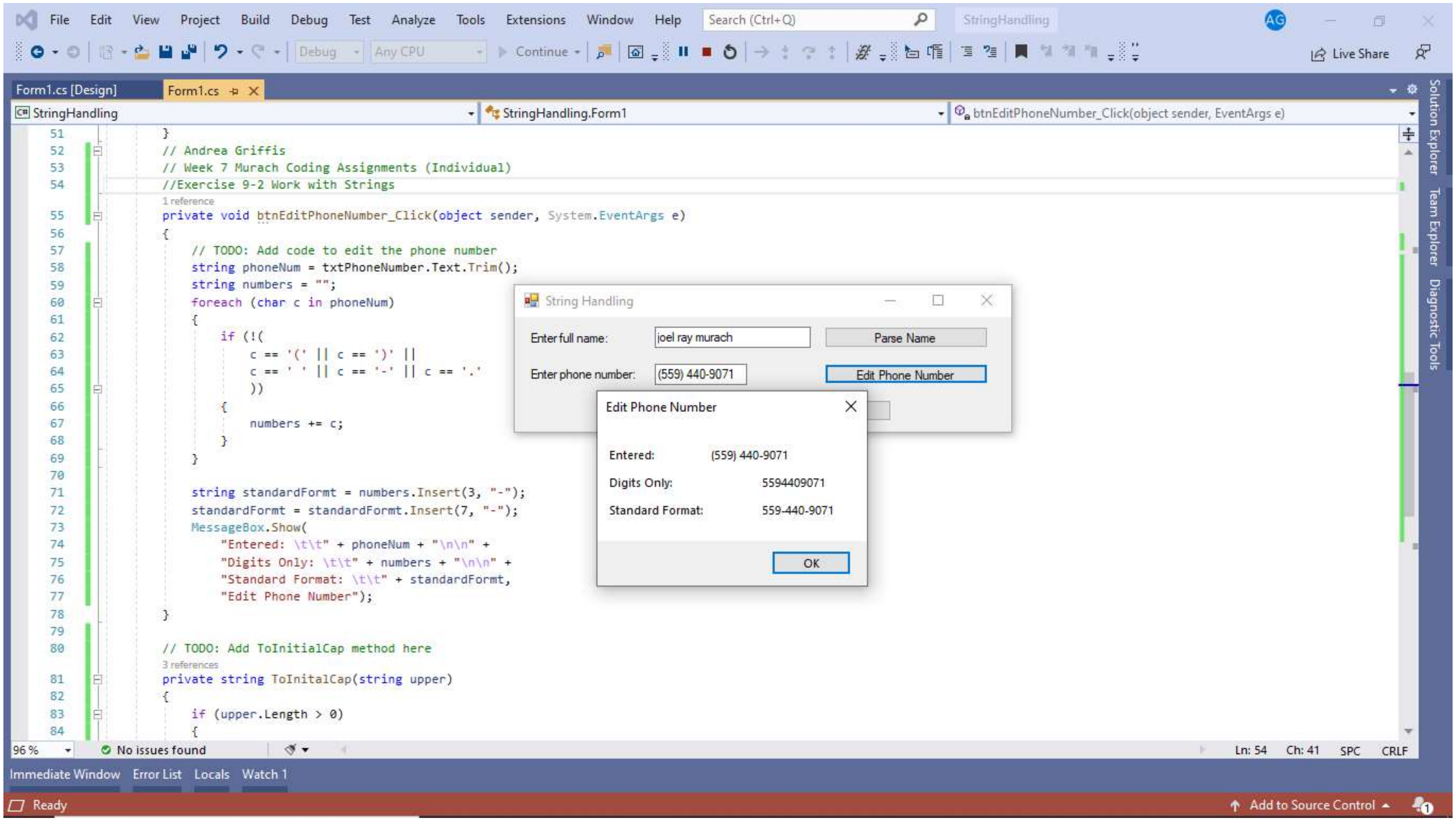
```
First Name:     Andrea
Middle Name:    Crystal
Last Name:      Griffis
```

Add code to edit a phone number



```
        }
        // Andrea Griffis
        // Week 7 Murach Coding Assignments (Individual)
        //Exercise 9-2 Work with Strings
        1 reference
        private void btnEditPhoneNumber_Click(object sender, System.EventArgs e)
        {
            // TODO: Add code to edit the phone number
            string phoneNum = txtPhoneNumber.Text.Trim();
            string numbers = "";
            foreach (char c in phoneNum)
            {
                if (!(
                    c == '(' || c == ')' ||
                    c == ' ' || c == '-' || c == '.'
                    ))
                {
                    numbers += c;
                }
            }

            string standardFormt = numbers.Insert(3, "-");
            standardFormt = standardFormt.Insert(7, "-");
            MessageBox.Show(
                "Entered: \t\t" + phoneNum + "\n\n" +
                "Digits Only: \t\t" + numbers + "\n\n" +
                "Standard Format: \t\t" + standardFormt,
                "Edit Phone Number");
        }

        // TODO: Add ToInitialCap method here
        3 references
        private string ToInitalCap(string upper)
        {
            if (upper.Length > 0)
            {
```

Test the application



```
        }
        // Andrea Griffis
        // Week 7 Murach Coding Assignments (Individual)
        //Exercise 9-2 Work with Strings
        1 reference
        private void btnEditPhoneNumber_Click(object sender, System.EventArgs e)
        {
            // TODO: Add code to edit the phone number
            string phoneNum = txtPhoneNumber.Text.Trim();
            string numbers = "";
            foreach (char c in phoneNum)
            {
                if (!(
                    c == '(' || c == ')' ||
                    c == ' ' || c == '-' || c == '.'
                    ))
                {
                    numbers += c;
                }
            }

            string standardFormt = numbers.Insert(3, "-");
            standardFormt = standardFormt.Insert(7, "-");
            MessageBox.Show(
                "Entered: \t\t" + phoneNum + "\n\n" +
                "Digits Only: \t\t" + numbers + "\n\n" +
                "Standard Format: \t\t" + standardFormt,
                "Edit Phone Number");
        }

        // TODO: Add ToInitialCap method here
        3 references
        private string ToInitalCap(string upper)
        {
            if (upper.Length > 0)
            {
```

String Handling

Enter full name:	joel ray murach	Parse Name
Enter phone number:	(559) 440-9071	Edit Phone Number

Edit Phone Number

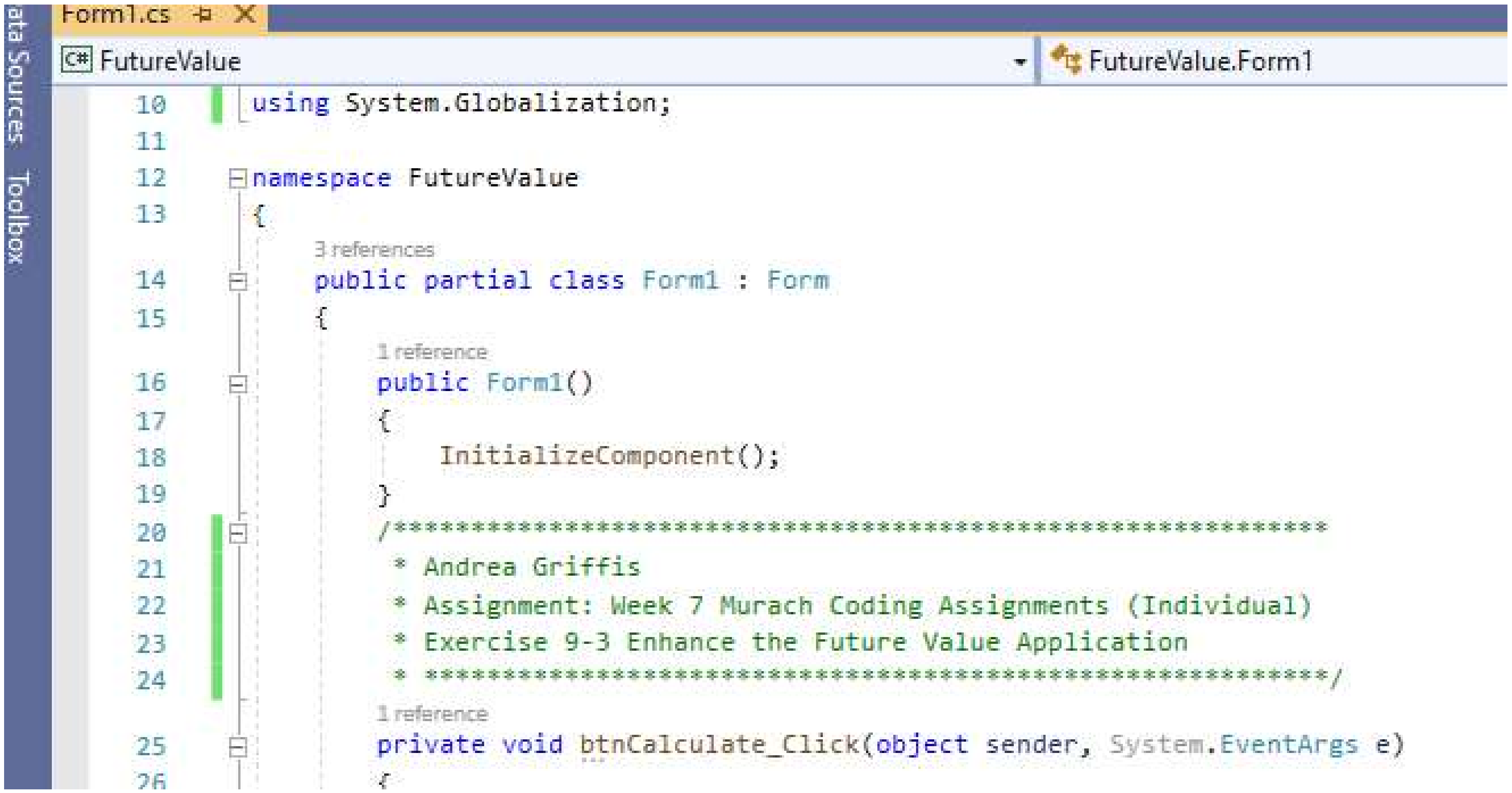
```
Entered:            (559) 440-9071

Digits Only:        5594409071

Standard Format:    559-440-9071
```

OK

Exercise 9-3 Enhance the Future Value application

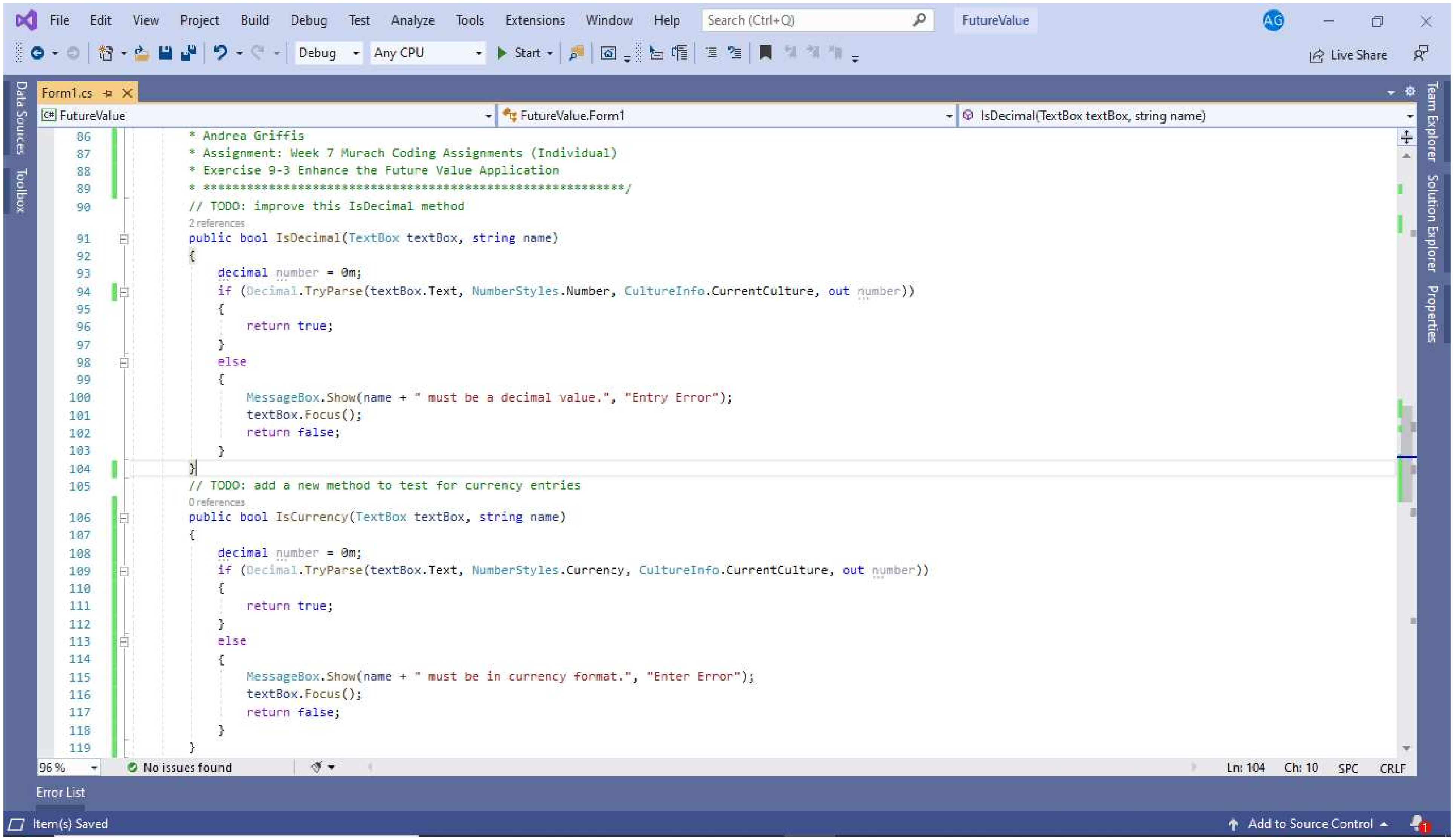
Add code that provides for formatted enteries (#'s 2- 10)

#2



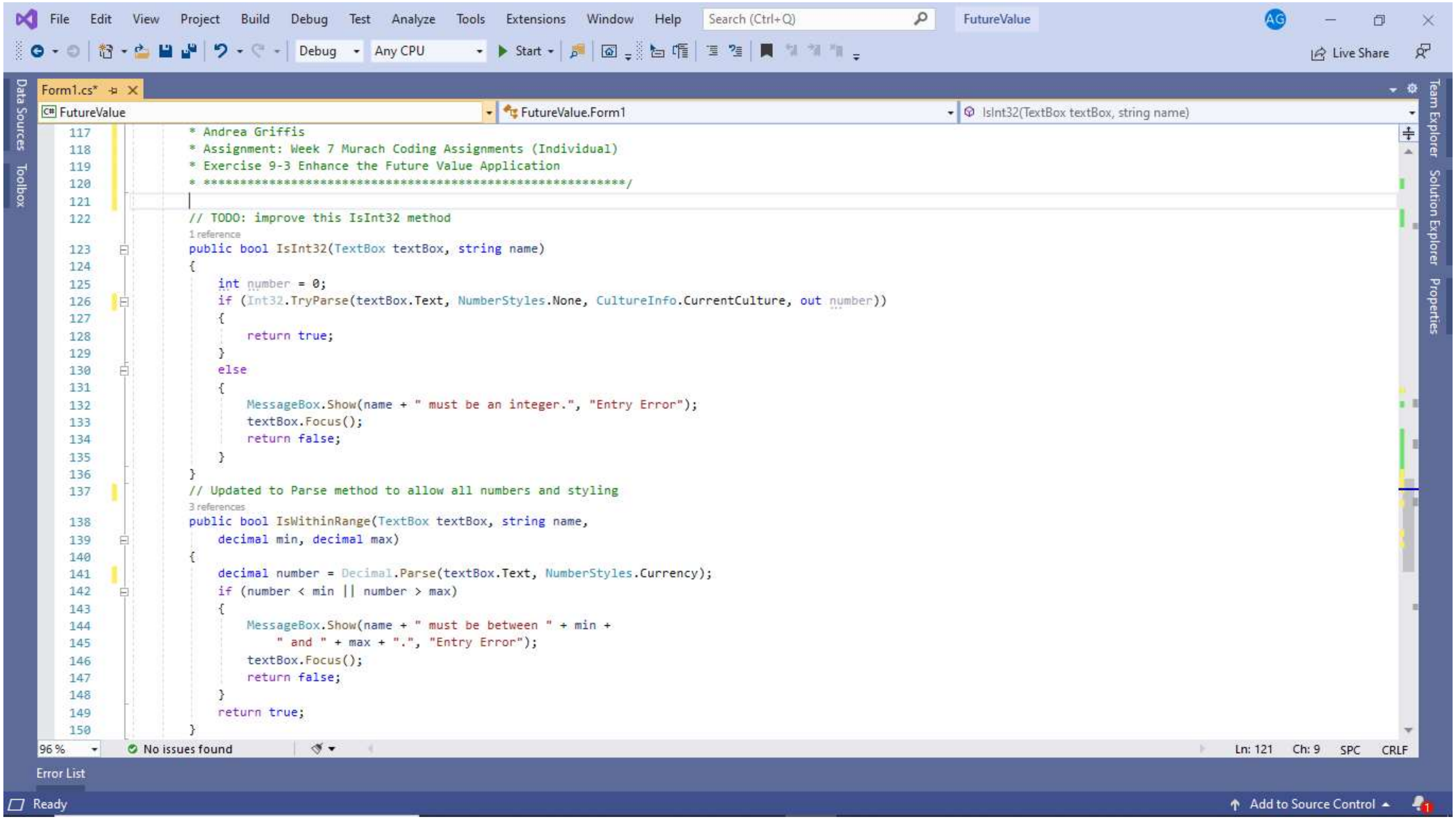
```
using System.Globalization;

namespace FutureValue
{
    3 references
    public partial class Form1 : Form
    {
        1 reference
        public Form1()
        {
            InitializeComponent();
        }
        /****************************************************************
         * Andrea Griffis
         * Assignment: Week 7 Murach Coding Assignments (Individual)
         * Exercise 9-3 Enhance the Future Value Application
         * **************************************************************/
        1 reference
        private void btnCalculate_Click(object sender, System.EventArgs e)
        {
```

#3 and #4



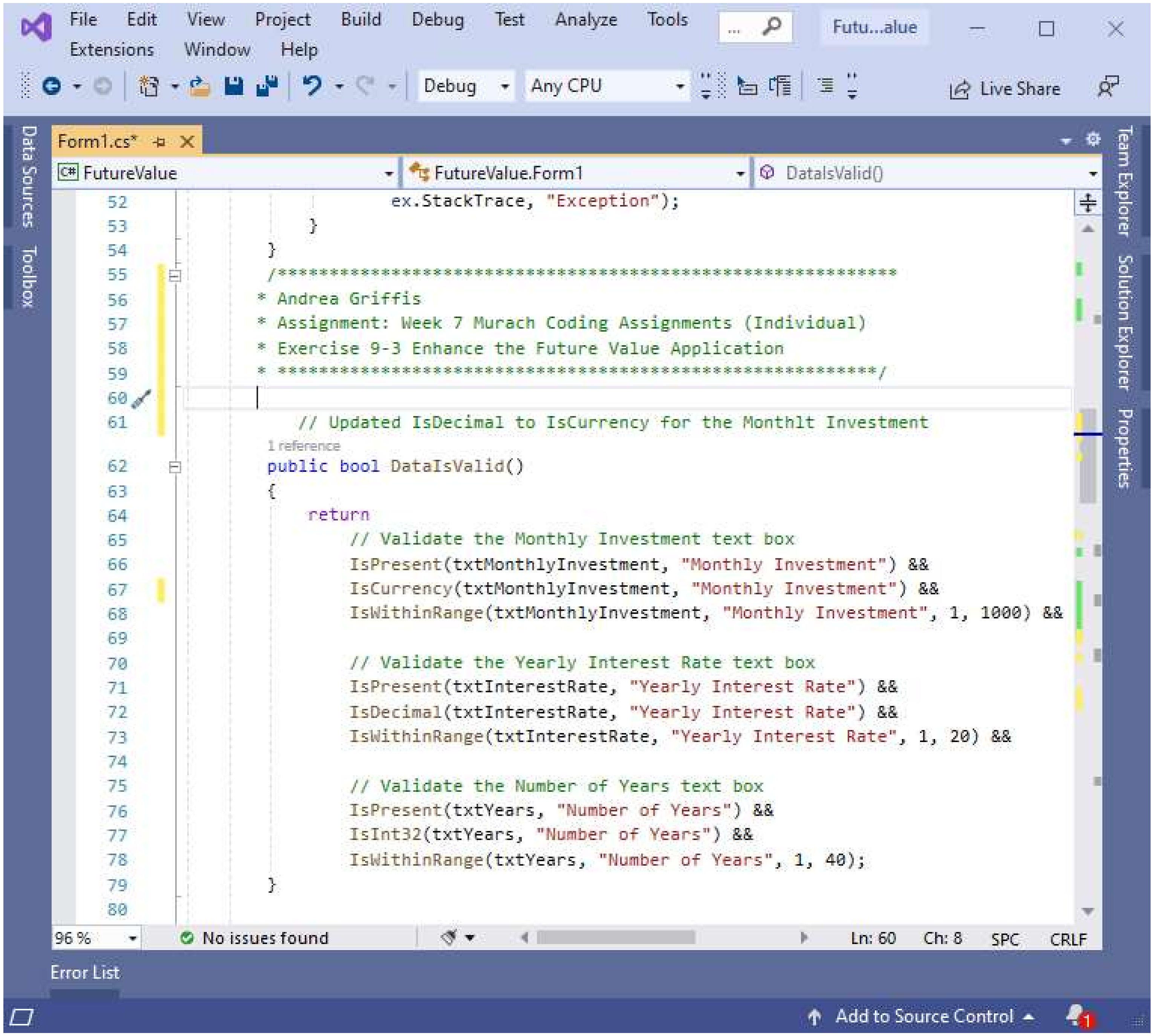
```
         * Andrea Griffis
         * Assignment: Week 7 Murach Coding Assignments (Individual)
         * Exercise 9-3 Enhance the Future Value Application
         * ***********************************************************/
        // TODO: improve this IsDecimal method
        public bool IsDecimal(TextBox textBox, string name)
        {
            decimal number = 0m;
            if (Decimal.TryParse(textBox.Text, NumberStyles.Number, CultureInfo.CurrentCulture, out number))
            {
                return true;
            }
            else
            {
                MessageBox.Show(name + " must be a decimal value.", "Entry Error");
                textBox.Focus();
                return false;
            }
        }
        // TODO: add a new method to test for currency entries
        public bool IsCurrency(TextBox textBox, string name)
        {
            decimal number = 0m;
            if (Decimal.TryParse(textBox.Text, NumberStyles.Currency, CultureInfo.CurrentCulture, out number))
            {
                return true;
            }
            else
            {
                MessageBox.Show(name + " must be in currency format.", "Enter Error");
                textBox.Focus();
                return false;
            }
        }
```

#5 and #6



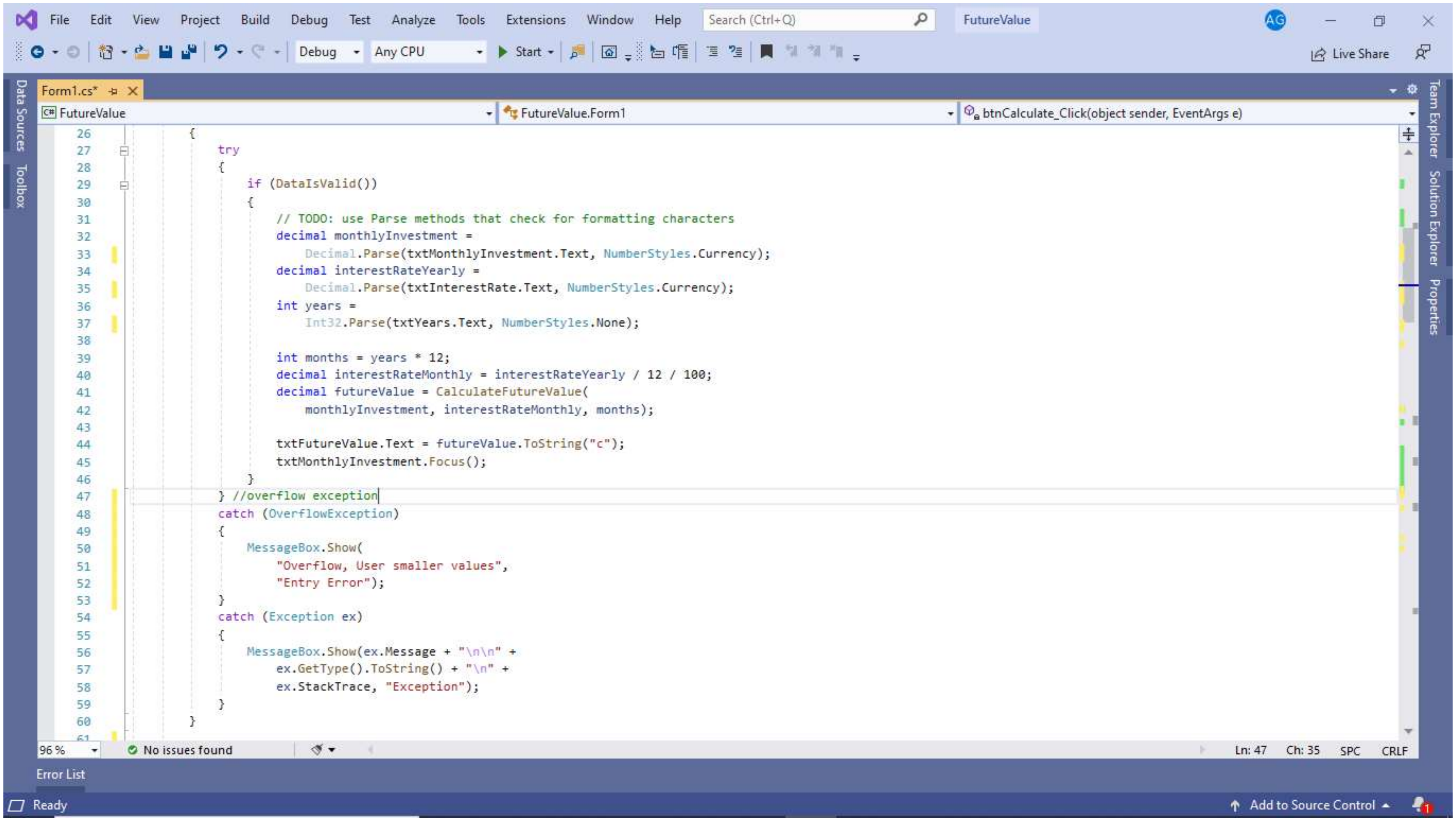
```
        * Andrea Griffis
        * Assignment: Week 7 Murach Coding Assignments (Individual)
        * Exercise 9-3 Enhance the Future Value Application
        * ************************************************************/

        // TODO: improve this IsInt32 method
        public bool IsInt32(TextBox textBox, string name)
        {
            int number = 0;
            if (Int32.TryParse(textBox.Text, NumberStyles.None, CultureInfo.CurrentCulture, out number))
            {
                return true;
            }
            else
            {
                MessageBox.Show(name + " must be an integer.", "Entry Error");
                textBox.Focus();
                return false;
            }
        }
        // Updated to Parse method to allow all numbers and styling
        public bool IsWithinRange(TextBox textBox, string name,
            decimal min, decimal max)
        {
            decimal number = Decimal.Parse(textBox.Text, NumberStyles.Currency);
            if (number < min || number > max)
            {
                MessageBox.Show(name + " must be between " + min +
                    " and " + max + ".", "Entry Error");
                textBox.Focus();
                return false;
            }
            return true;
        }
```

#7



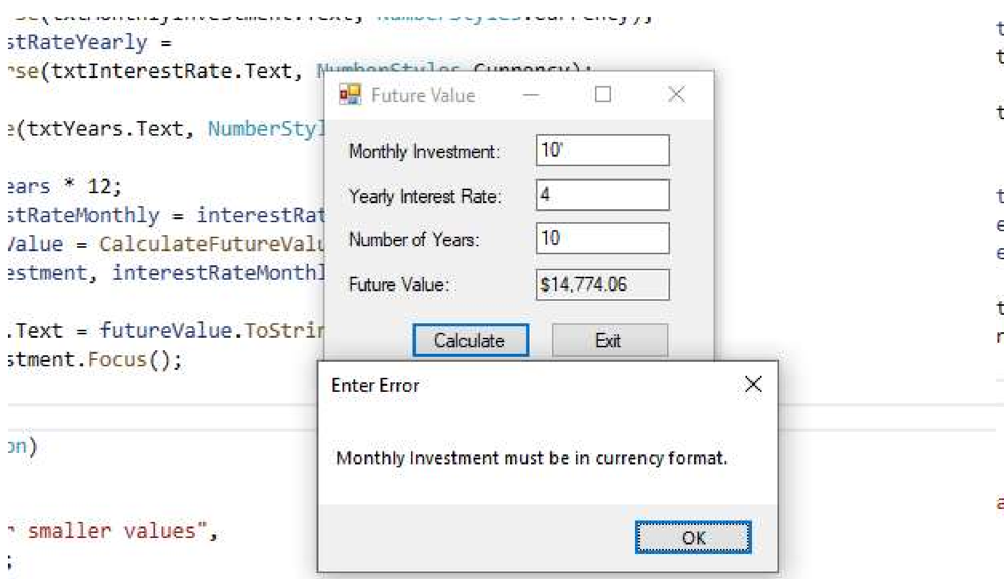
```
                    ex.StackTrace, "Exception");
            }
        }
        /*************************************************************
       * Andrea Griffis
       * Assignment: Week 7 Murach Coding Assignments (Individual)
       * Exercise 9-3 Enhance the Future Value Application
       * ************************************************************/
       
            // Updated IsDecimal to IsCurrency for the Monthlt Investment
        1 reference
        public bool DataIsValid()
        {
            return
                // Validate the Monthly Investment text box
                IsPresent(txtMonthlyInvestment, "Monthly Investment") &&
                IsCurrency(txtMonthlyInvestment, "Monthly Investment") &&
                IsWithinRange(txtMonthlyInvestment, "Monthly Investment", 1, 1000) &&

                // Validate the Yearly Interest Rate text box
                IsPresent(txtInterestRate, "Yearly Interest Rate") &&
                IsDecimal(txtInterestRate, "Yearly Interest Rate") &&
                IsWithinRange(txtInterestRate, "Yearly Interest Rate", 1, 20) &&

                // Validate the Number of Years text box
                IsPresent(txtYears, "Number of Years") &&
                IsInt32(txtYears, "Number of Years") &&
                IsWithinRange(txtYears, "Number of Years", 1, 40);
        }

```

#8 and 9



```
        {
            try
            {
                if (DataIsValid())
                {
                    // TODO: use Parse methods that check for formatting characters
                    decimal monthlyInvestment =
                        Decimal.Parse(txtMonthlyInvestment.Text, NumberStyles.Currency);
                    decimal interestRateYearly =
                        Decimal.Parse(txtInterestRate.Text, NumberStyles.Currency);
                    int years =
                        Int32.Parse(txtYears.Text, NumberStyles.None);

                    int months = years * 12;
                    decimal interestRateMonthly = interestRateYearly / 12 / 100;
                    decimal futureValue = CalculateFutureValue(
                        monthlyInvestment, interestRateMonthly, months);

                    txtFutureValue.Text = futureValue.ToString("c");
                    txtMonthlyInvestment.Focus();
                }
            } //overflow exception
            catch (OverflowException)
            {
                MessageBox.Show(
                    "Overflow, User smaller values",
                    "Entry Error");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message + "\n\n" +
                    ex.GetType().ToString() + "\n" +
                    ex.StackTrace, "Exception");
            }
        }
```

#10 Test the application

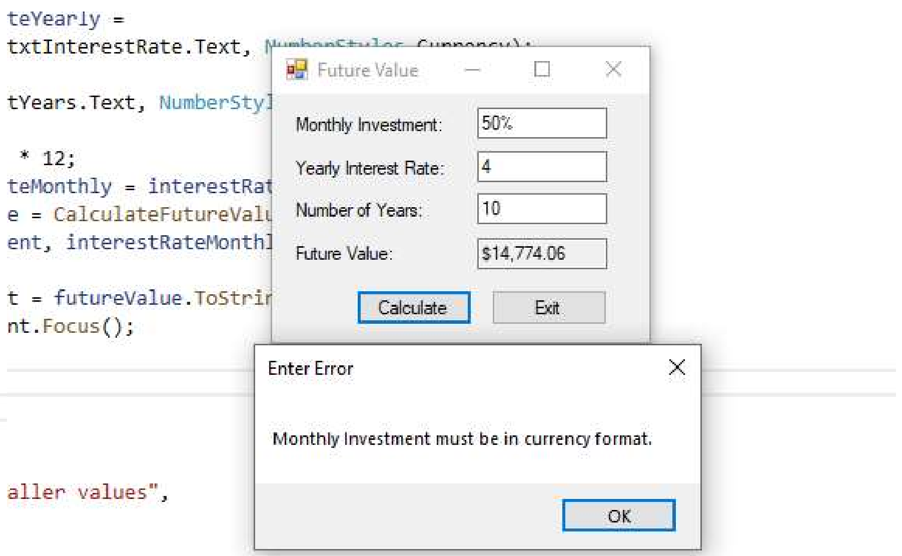


```
stRateYearly =
rse(txtInterestRate.Text, NumberStyles.Currency);

e(txtYears.Text, NumberSty

ears * 12;
stRateMonthly = interestRat
Value = CalculateFutureValu
estment, interestRateMonthl

.Text = futureValue.ToStrir
stment.Focus();

on)

 smaller values",
;
```

Monthly Investment: 10'
Yearly Interest Rate: 4
Number of Years: 10
Future Value: $14,774.06

Enter Error

Monthly Investment must be in currency format.



```
teYearly =
txtInterestRate.Text, NumberStyles.Currency);

tYears.Text, NumberSty

 * 12;
teMonthly = interestRat
e = CalculateFutureValu
ent, interestRateMonthl

t = futureValue.ToStrir
nt.Focus();

aller values",
```

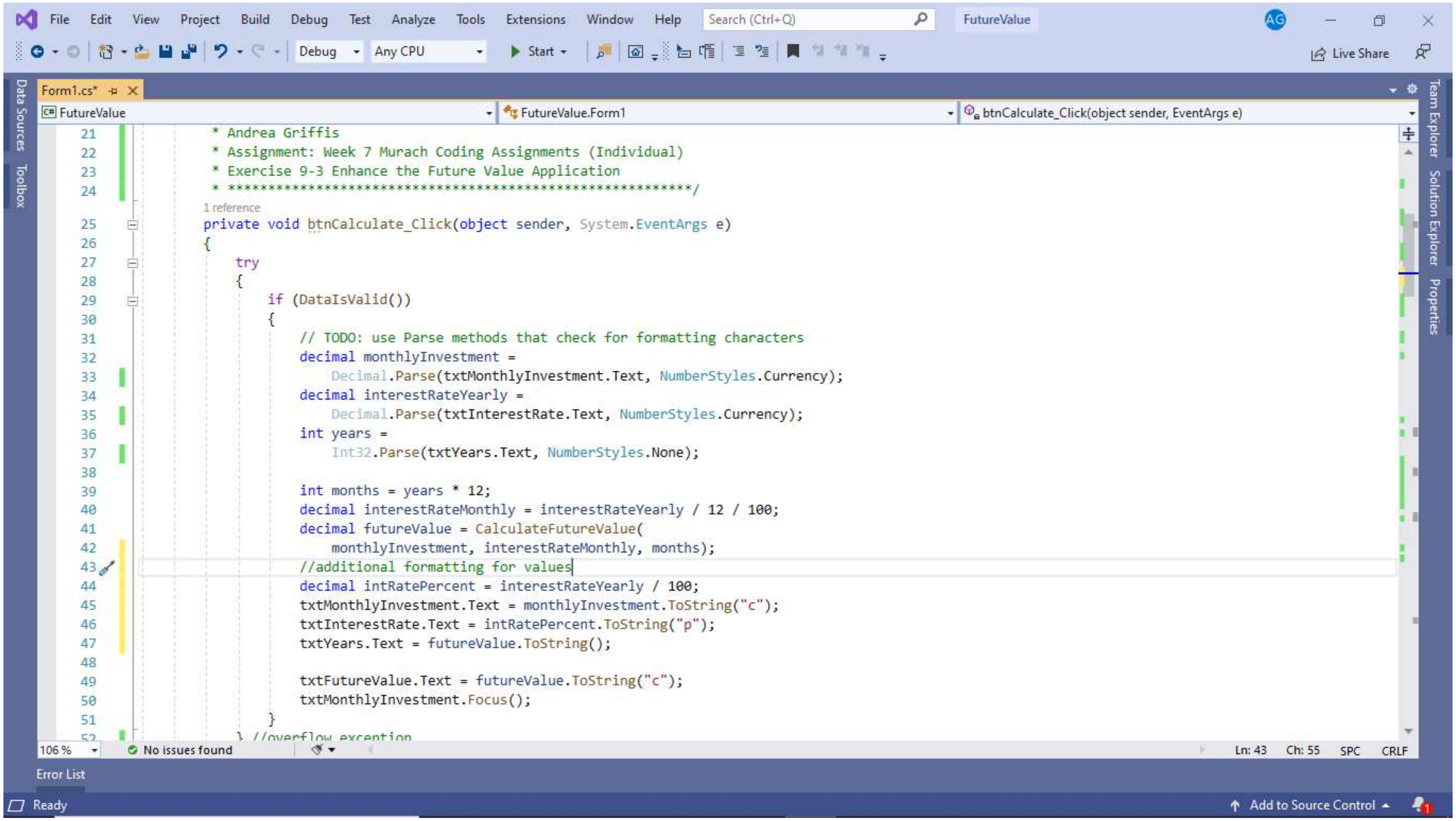
Monthly Investment: 50%
Yearly Interest Rate: 4
Number of Years: 10
Future Value: $14,774.06

Enter Error

Monthly Investment must be in currency format.

Add code to format the displayed values

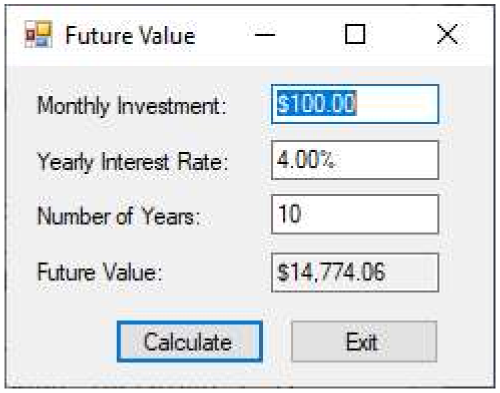
#11



```
         * Andrea Griffis
         * Assignment: Week 7 Murach Coding Assignments (Individual)
         * Exercise 9-3 Enhance the Future Value Application
         * ************************************************************/
        1 reference
        private void btnCalculate_Click(object sender, System.EventArgs e)
        {
            try
            {
                if (DataIsValid())
                {
                    // TODO: use Parse methods that check for formatting characters
                    decimal monthlyInvestment =
                        Decimal.Parse(txtMonthlyInvestment.Text, NumberStyles.Currency);
                    decimal interestRateYearly =
                        Decimal.Parse(txtInterestRate.Text, NumberStyles.Currency);
                    int years =
                        Int32.Parse(txtYears.Text, NumberStyles.None);

                    int months = years * 12;
                    decimal interestRateMonthly = interestRateYearly / 12 / 100;
                    decimal futureValue = CalculateFutureValue(
                        monthlyInvestment, interestRateMonthly, months);
                    //additional formatting for values
                    decimal intRatePercent = interestRateYearly / 100;
                    txtMonthlyInvestment.Text = monthlyInvestment.ToString("c");
                    txtInterestRate.Text = intRatePercent.ToString("p");
                    txtYears.Text = futureValue.ToString();

                    txtFutureValue.Text = futureValue.ToString("c");
                    txtMonthlyInvestment.Focus();
                }
            } //overflow exception
```

Test the application

When entered with only numbers


Future Value

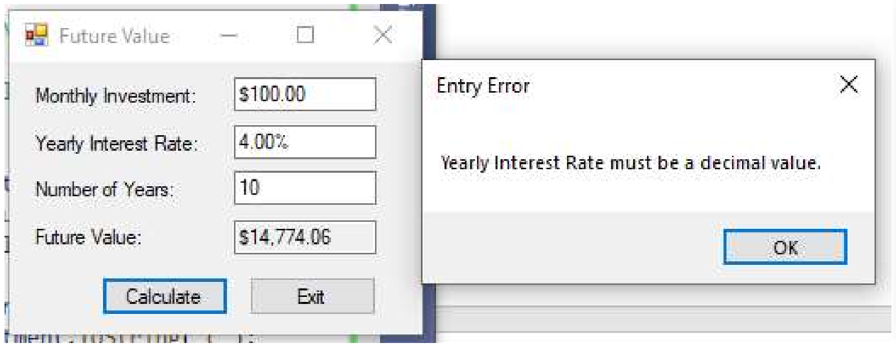
Monthly Investment: $100.00

Yearly Interest Rate: 4.00%

Number of Years: 10

Future Value: $14,774.06

When entered with % sign in interest rate and monthly investment


Future Value

Monthly Investment: $100.00

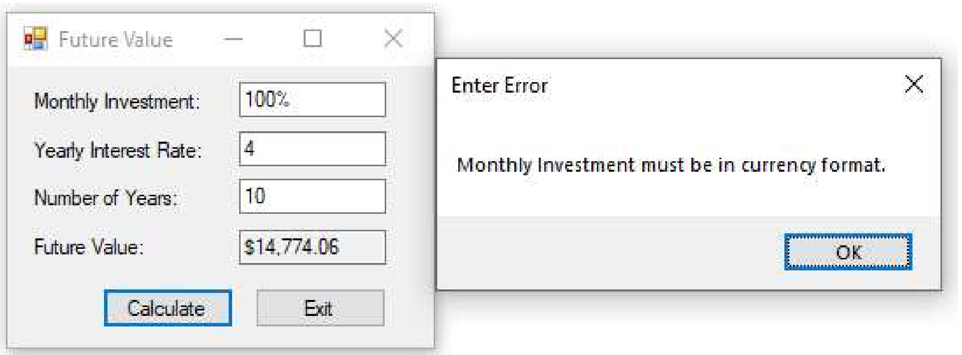
Yearly Interest Rate: 4.00%

Number of Years: 10

Future Value: $14,774.06

Entry Error

Yearly Interest Rate must be a decimal value.


Future Value

Monthly Investment: 100%

Yearly Interest Rate: 4

Number of Years: 10

Future Value: $14,774.06

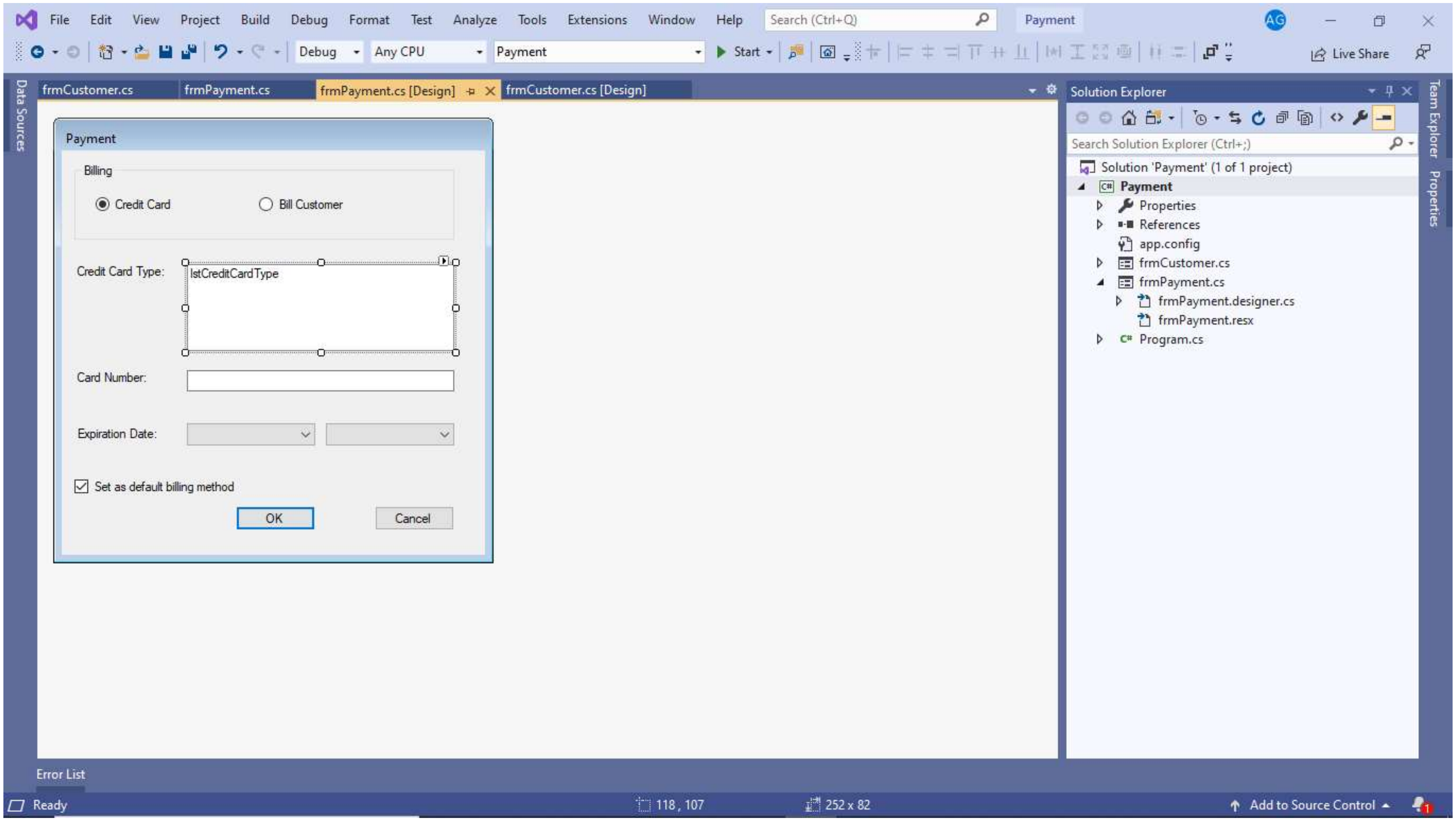
Enter Error

Monthly Investment must be in currency format.

Exercise 10-1 Create the Payment application

Add code that provides for formatted enteries (#'s 2- 10)

#1-3 Open the project and prepare 2 forms



Payment

Billing

(● Credit Card) ○ Bill Customer

Credit Card Type: lstCreditCardType

Card Number:

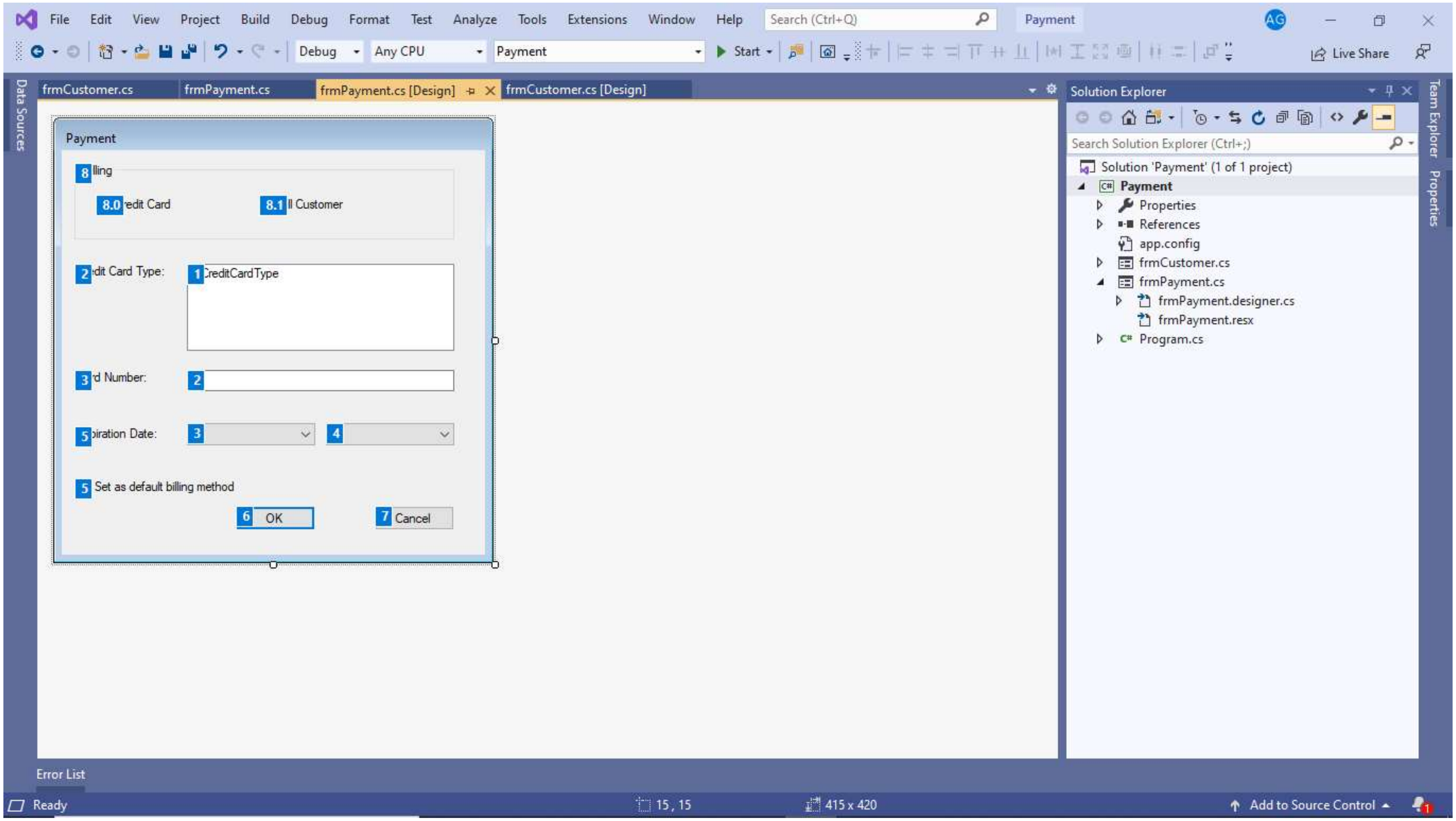
Expiration Date:

☑ Set as default billing method

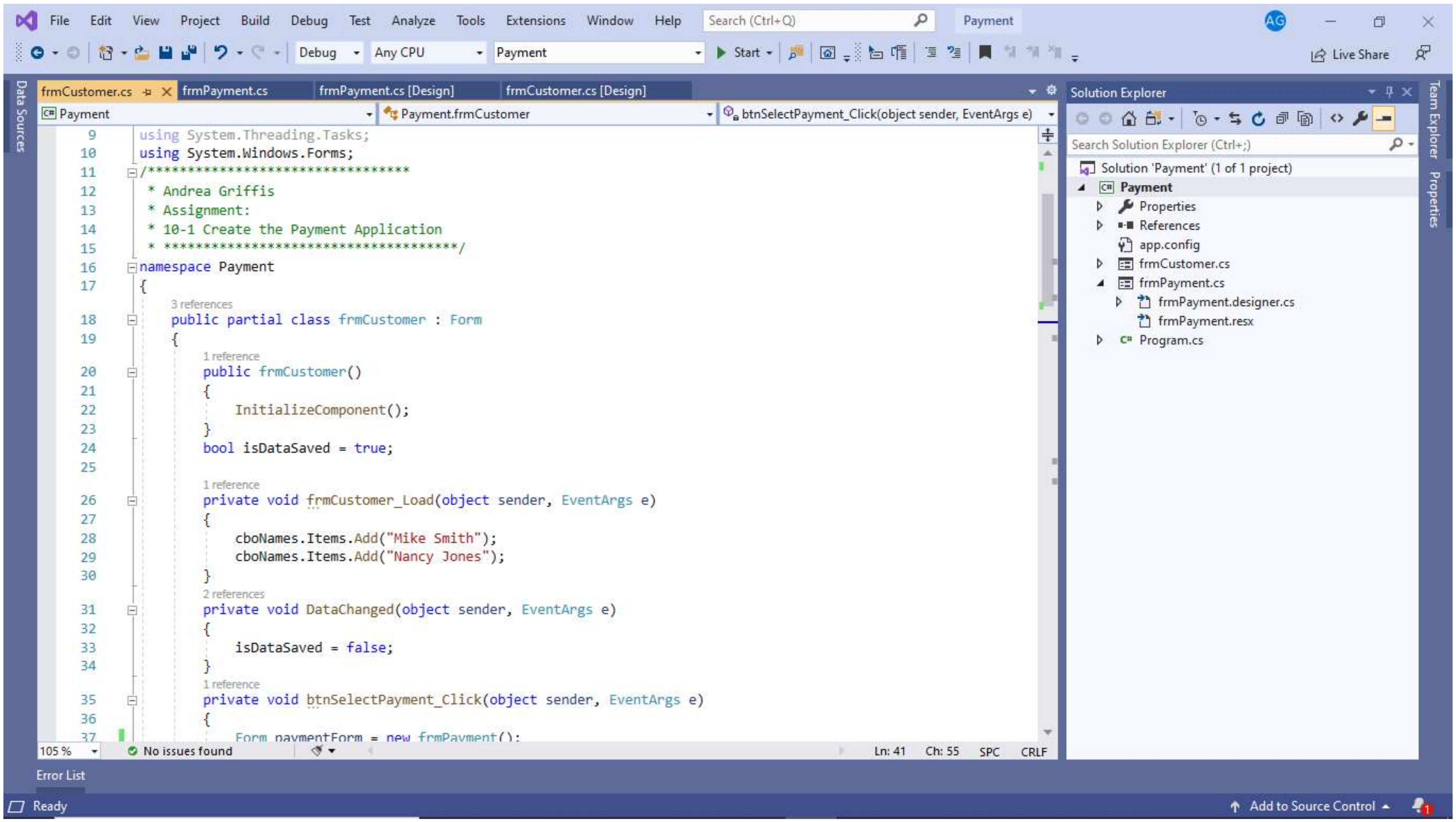
OK Cancel

Solution 'Payment' (1 of 1 project)
- Payment
  - Properties
  - References
  - app.config
  - frmCustomer.cs
  - frmPayment.cs
    - frmPayment.designer.cs
    - frmPayment.resx
  - Program.cs

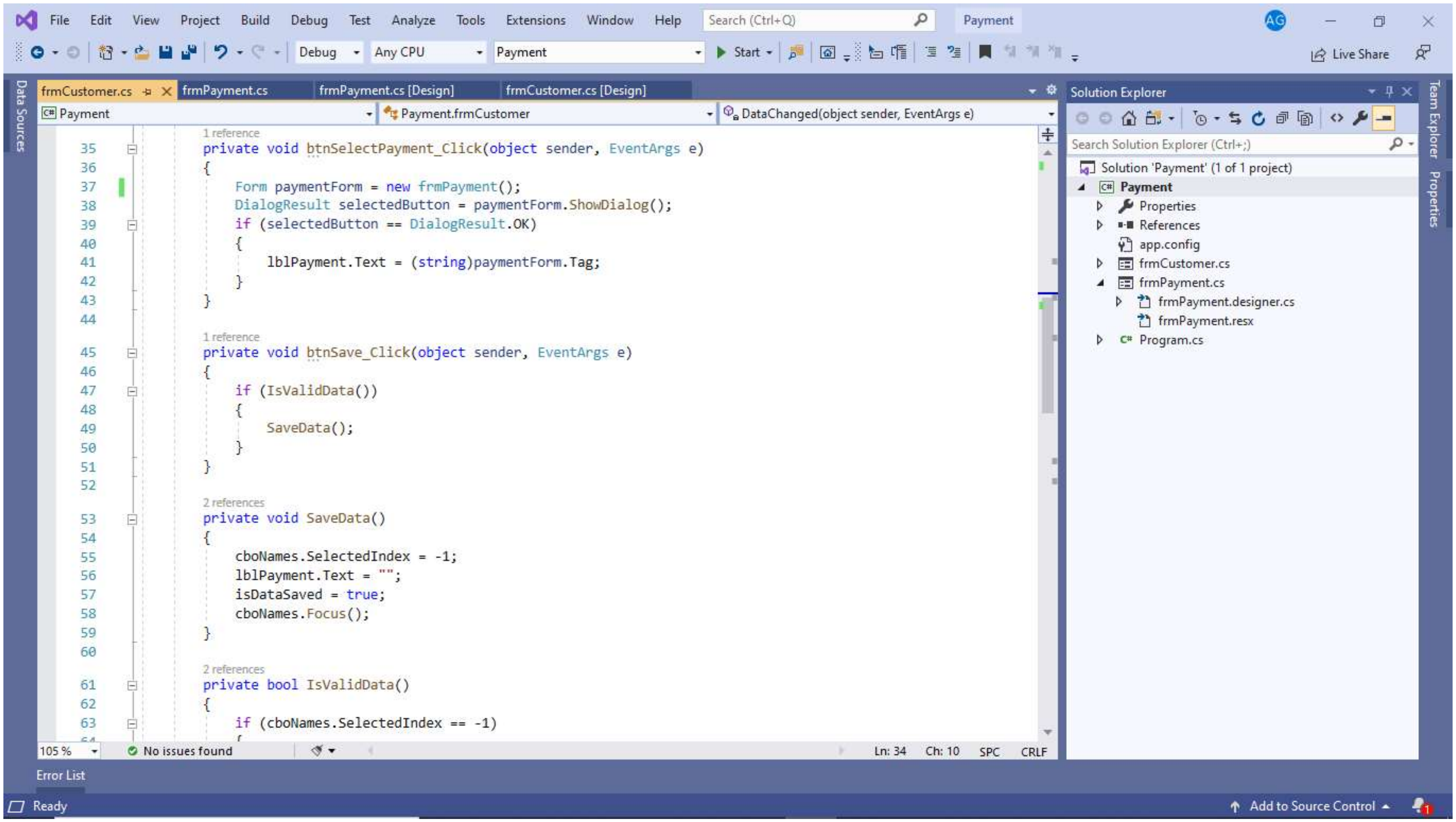
#4&5 Design the payment form



Add the code for the Customer form #6-9



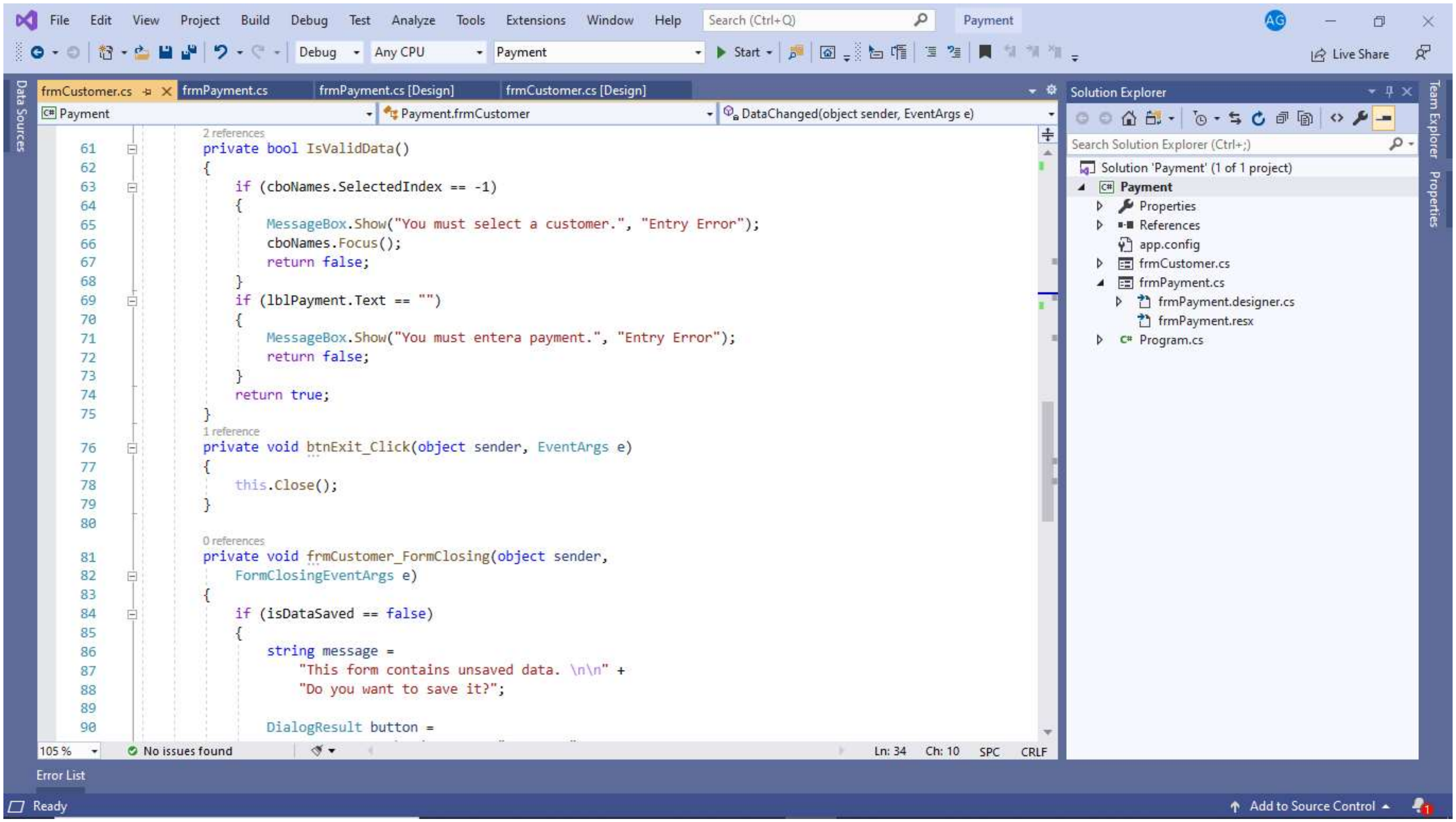
```
using System.Threading.Tasks;
using System.Windows.Forms;
/*********************************
 * Andrea Griffis
 * Assignment:
 * 10-1 Create the Payment Application
 * ***********************************/
namespace Payment
{
    3 references
    public partial class frmCustomer : Form
    {
        1 reference
        public frmCustomer()
        {
            InitializeComponent();
        }
        bool isDataSaved = true;

        1 reference
        private void frmCustomer_Load(object sender, EventArgs e)
        {
            cboNames.Items.Add("Mike Smith");
            cboNames.Items.Add("Nancy Jones");
        }
        2 references
        private void DataChanged(object sender, EventArgs e)
        {
            isDataSaved = false;
        }
        1 reference
        private void btnSelectPayment_Click(object sender, EventArgs e)
        {
            Form paymentForm = new frmPayment();
```



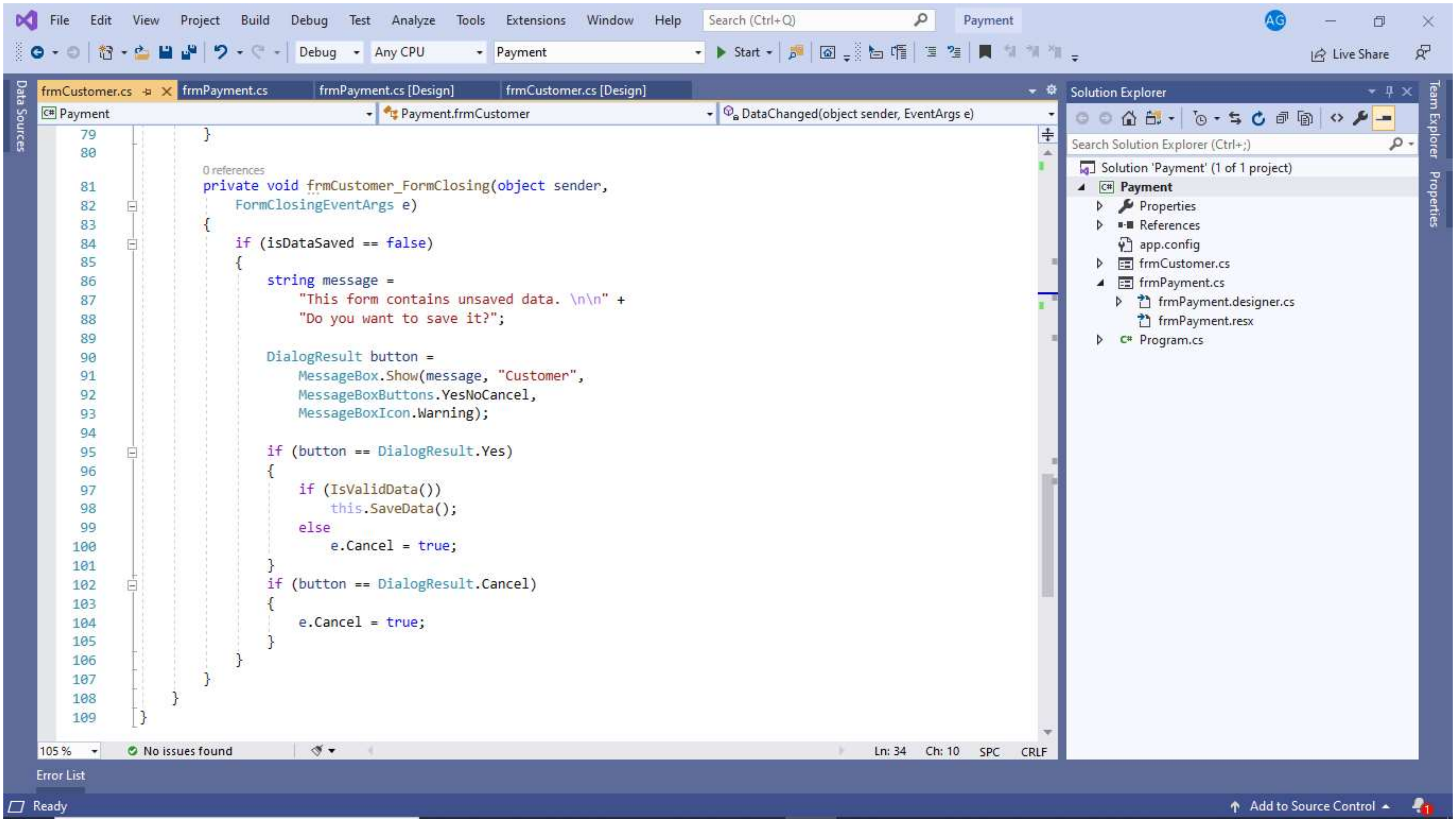
```
        1 reference
        private void btnSelectPayment_Click(object sender, EventArgs e)
        {
            Form paymentForm = new frmPayment();
            DialogResult selectedButton = paymentForm.ShowDialog();
            if (selectedButton == DialogResult.OK)
            {
                lblPayment.Text = (string)paymentForm.Tag;
            }
        }

        1 reference
        private void btnSave_Click(object sender, EventArgs e)
        {
            if (IsValidData())
            {
                SaveData();
            }
        }

        2 references
        private void SaveData()
        {
            cboNames.SelectedIndex = -1;
            lblPayment.Text = "";
            isDataSaved = true;
            cboNames.Focus();
        }

        2 references
        private bool IsValidData()
        {
            if (cboNames.SelectedIndex == -1)
```

Solution Explorer

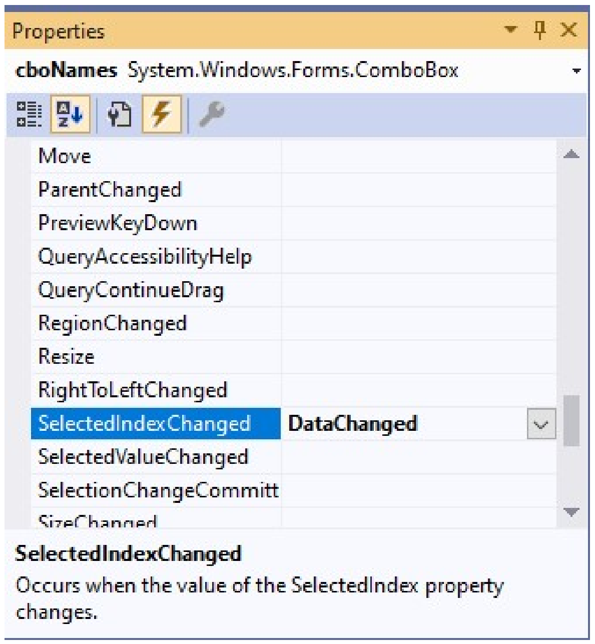
Solution 'Payment' (1 of 1 project)
- Payment
  - Properties
  - References
  - app.config
  - frmCustomer.cs
  - frmPayment.cs
    - frmPayment.designer.cs
    - frmPayment.resx
  - Program.cs



```
2 references
private bool IsValidData()
{
    if (cboNames.SelectedIndex == -1)
    {
        MessageBox.Show("You must select a customer.", "Entry Error");
        cboNames.Focus();
        return false;
    }
    if (lblPayment.Text == "")
    {
        MessageBox.Show("You must entera payment.", "Entry Error");
        return false;
    }
    return true;
}
1 reference
private void btnExit_Click(object sender, EventArgs e)
{
    this.Close();
}

0 references
private void frmCustomer_FormClosing(object sender,
    FormClosingEventArgs e)
{
    if (isDataSaved == false)
    {
        string message =
            "This form contains unsaved data. \n\n" +
            "Do you want to save it?";

        DialogResult button =
```



```
            }

            0 references
            private void frmCustomer_FormClosing(object sender,
                FormClosingEventArgs e)
            {
                if (isDataSaved == false)
                {
                    string message =
                        "This form contains unsaved data. \n\n" +
                        "Do you want to save it?";

                    DialogResult button =
                        MessageBox.Show(message, "Customer",
                        MessageBoxButtons.YesNoCancel,
                        MessageBoxIcon.Warning);

                    if (button == DialogResult.Yes)
                    {
                        if (IsValidData())
                            this.SaveData();
                        else
                            e.Cancel = true;
                    }
                    if (button == DialogResult.Cancel)
                    {
                        e.Cancel = true;
                    }
                }
            }
        }
    }
```

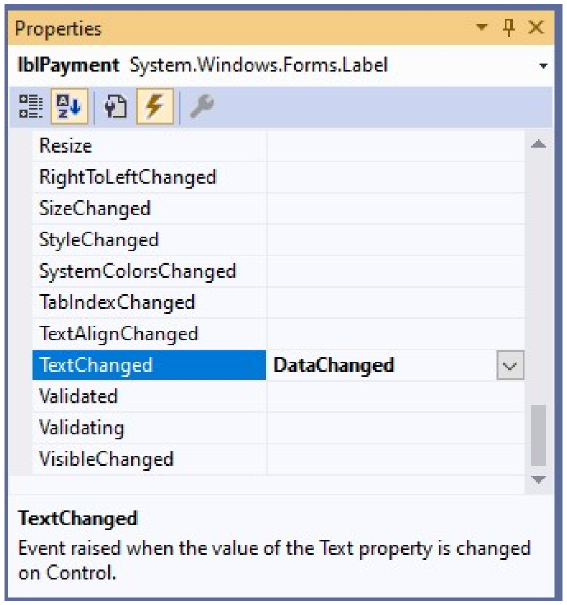


Properties

cboNames System.Windows.Forms.ComboBox

Move	
ParentChanged	
PreviewKeyDown	
QueryAccessibilityHelp	
QueryContinueDrag	
RegionChanged	
Resize	
RightToLeftChanged	
SelectedIndexChanged	DataChanged
SelectedValueChanged	
SelectionChangeCommitt	
SizeChanged	

SelectedIndexChanged
Occurs when the value of the SelectedIndex property changes.



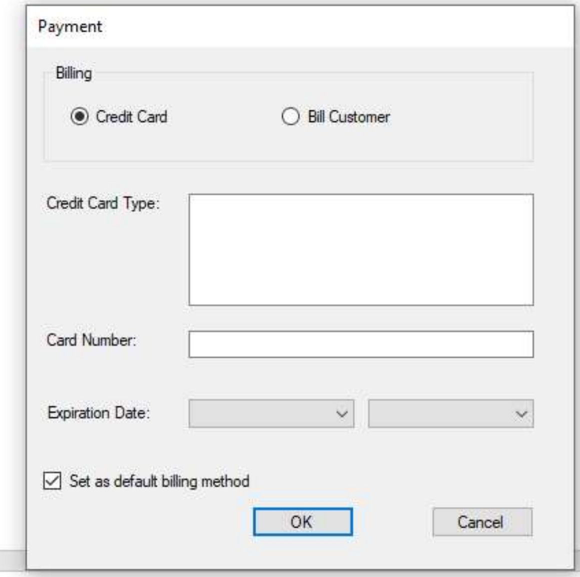
Properties

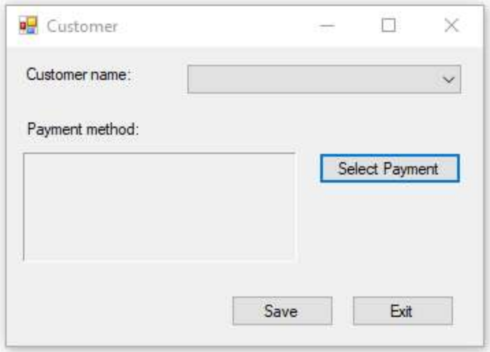
lblPayment System.Windows.Forms.Label

Resize	
RightToLeftChanged	
SizeChanged	
StyleChanged	
SystemColorsChanged	
TabIndexChanged	
TextAlignChanged	
TextChanged	DataChanged
Validated	
Validating	
VisibleChanged	

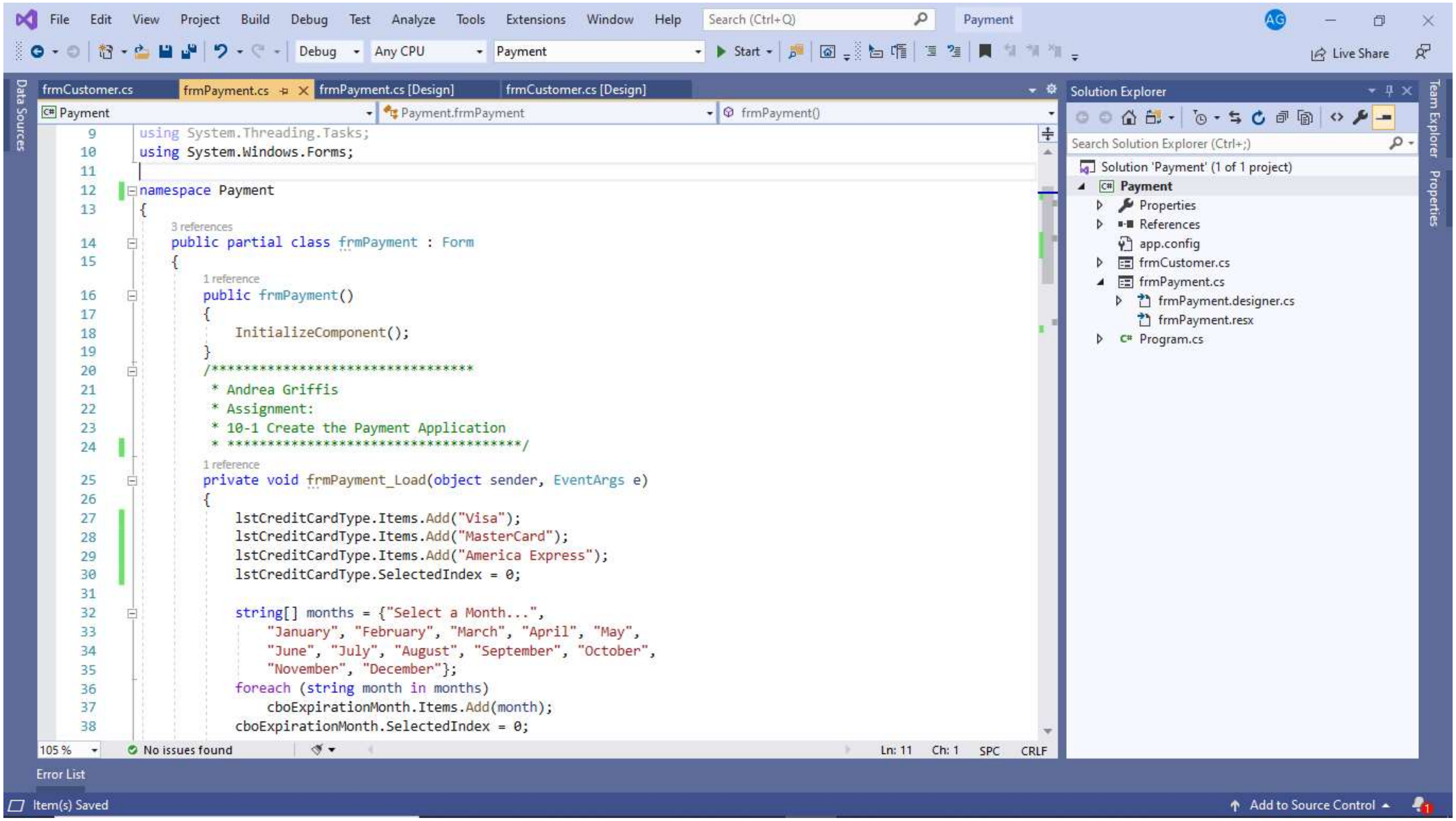
TextChanged
Event raised when the value of the Text property is changed on Control.

Test the application

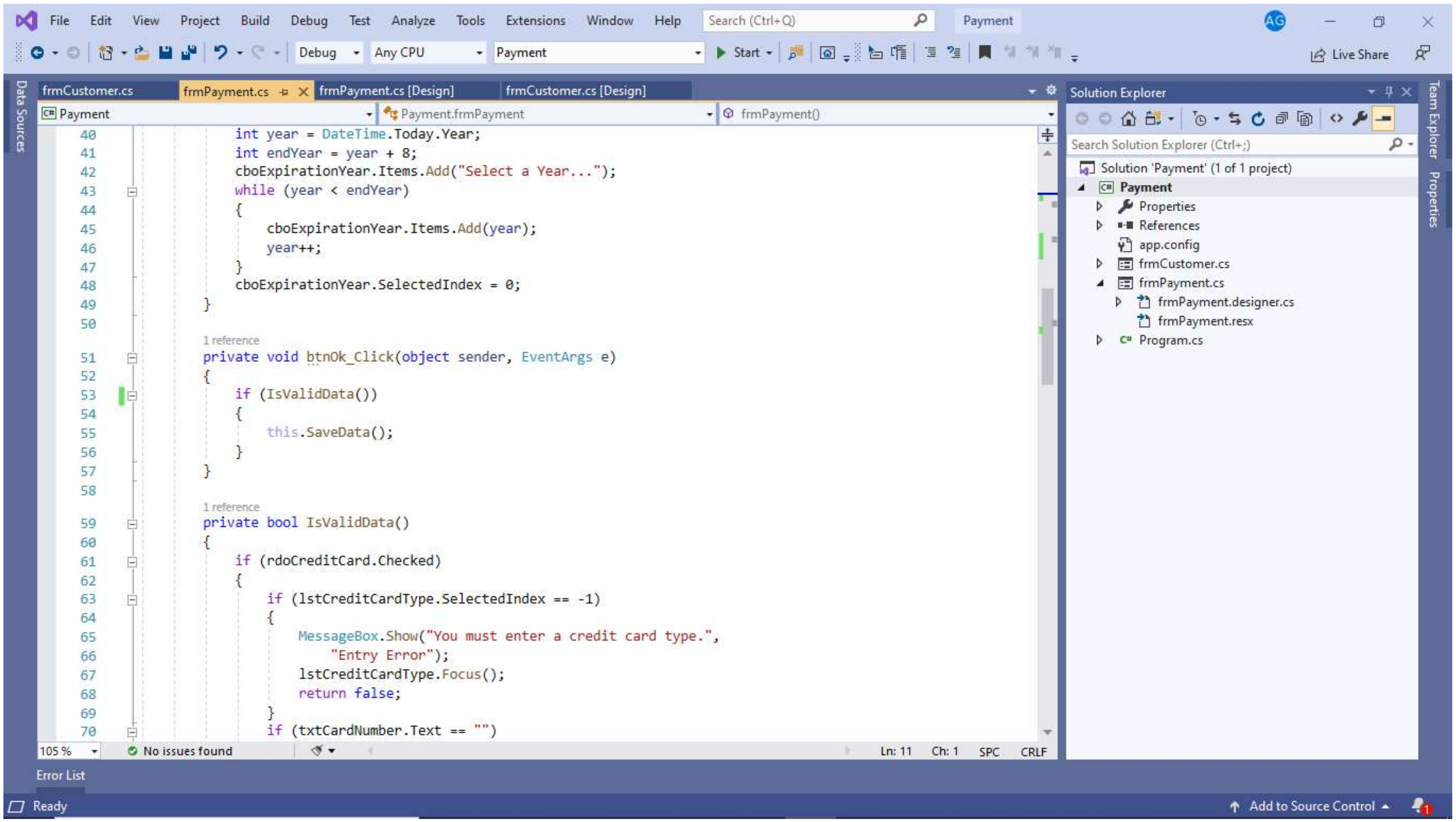




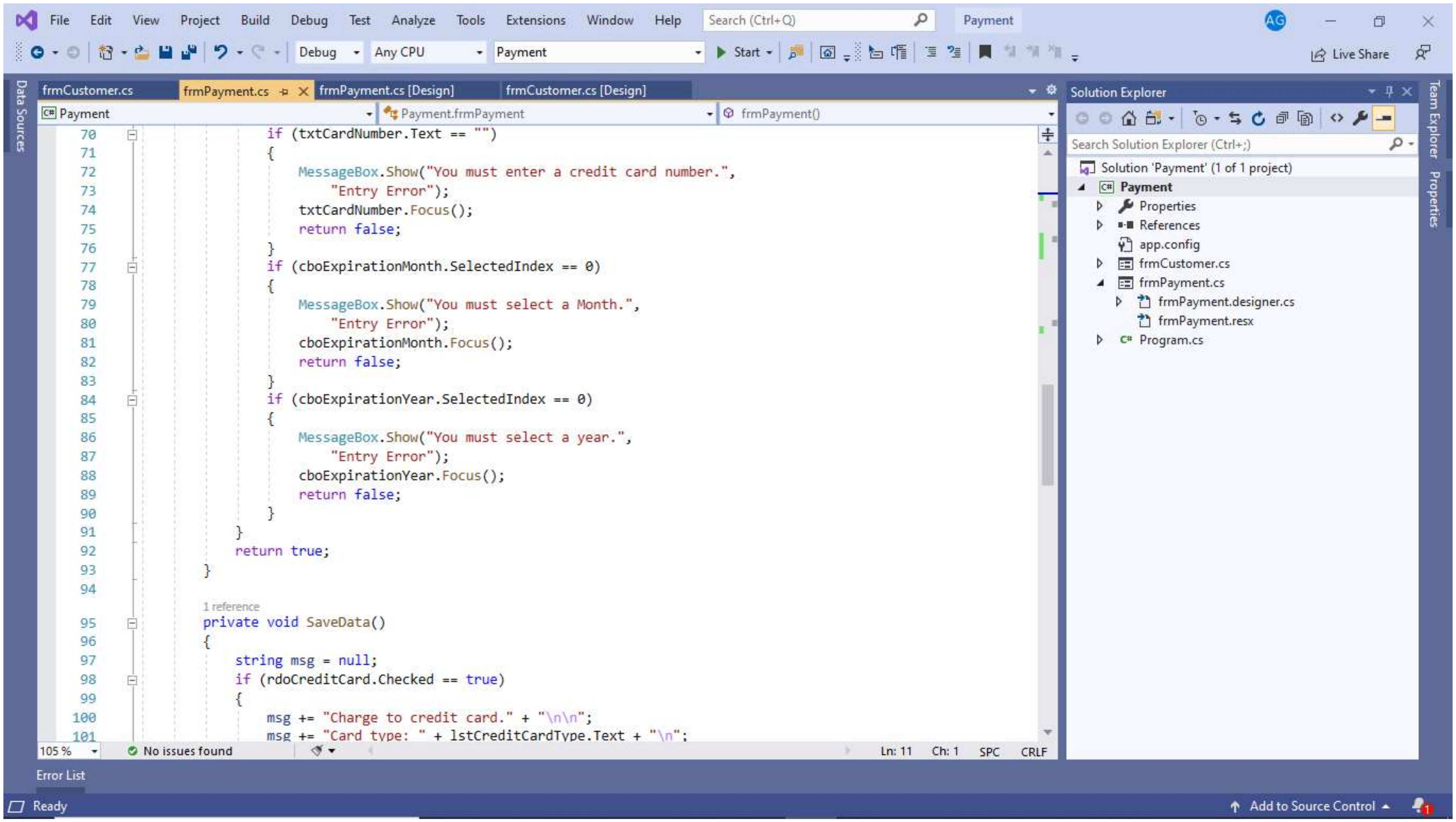
Add code to the Payment form #10-13



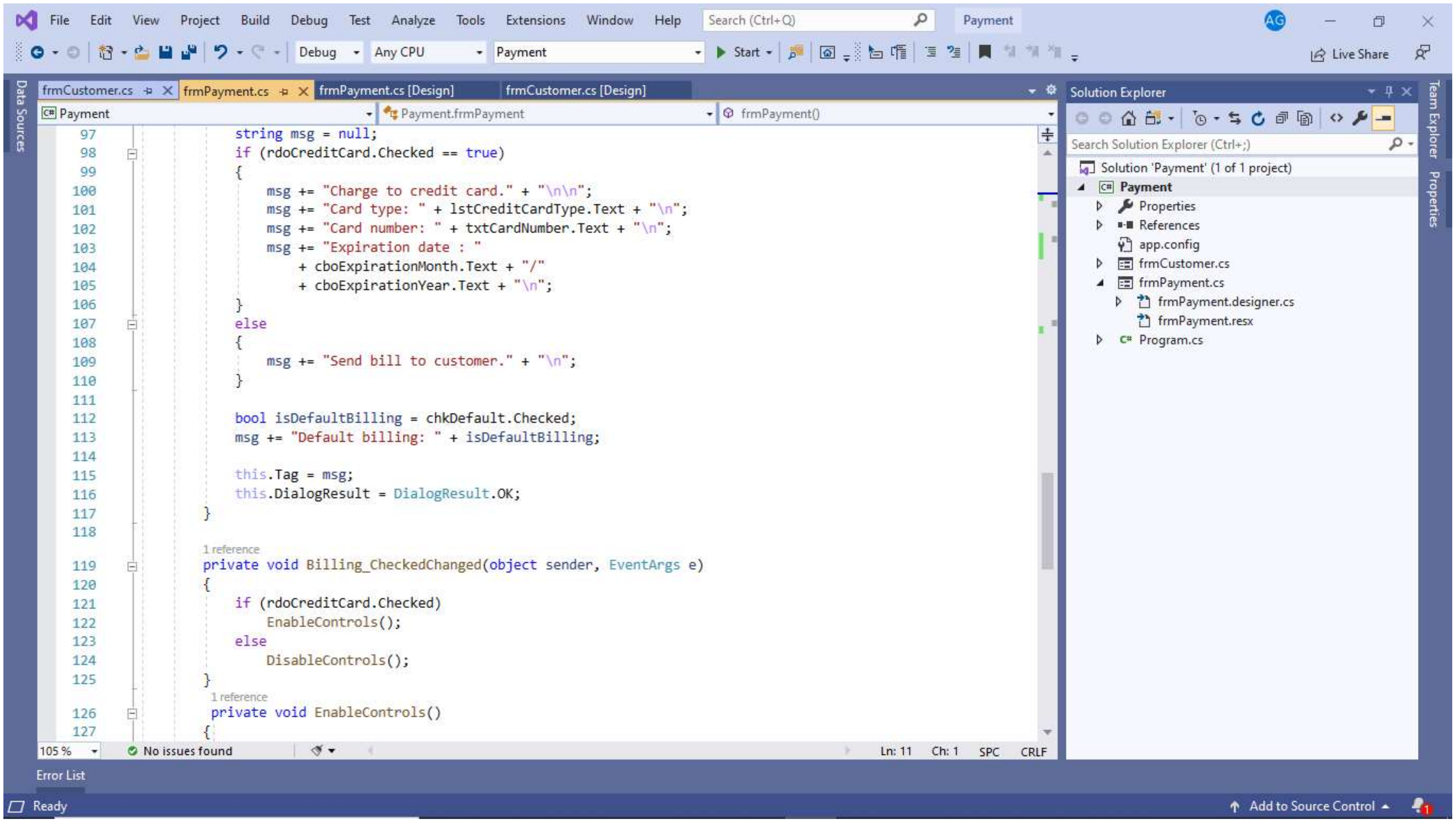
```
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Payment
{
    public partial class frmPayment : Form
    {
        public frmPayment()
        {
            InitializeComponent();
        }
        /*********************************
         * Andrea Griffis
         * Assignment:
         * 10-1 Create the Payment Application
         * *************************************/
        private void frmPayment_Load(object sender, EventArgs e)
        {
            lstCreditCardType.Items.Add("Visa");
            lstCreditCardType.Items.Add("MasterCard");
            lstCreditCardType.Items.Add("America Express");
            lstCreditCardType.SelectedIndex = 0;

            string[] months = {"Select a Month...",
                "January", "February", "March", "April", "May",
                "June", "July", "August", "September", "October",
                "November", "December"};
            foreach (string month in months)
                cboExpirationMonth.Items.Add(month);
            cboExpirationMonth.SelectedIndex = 0;
```



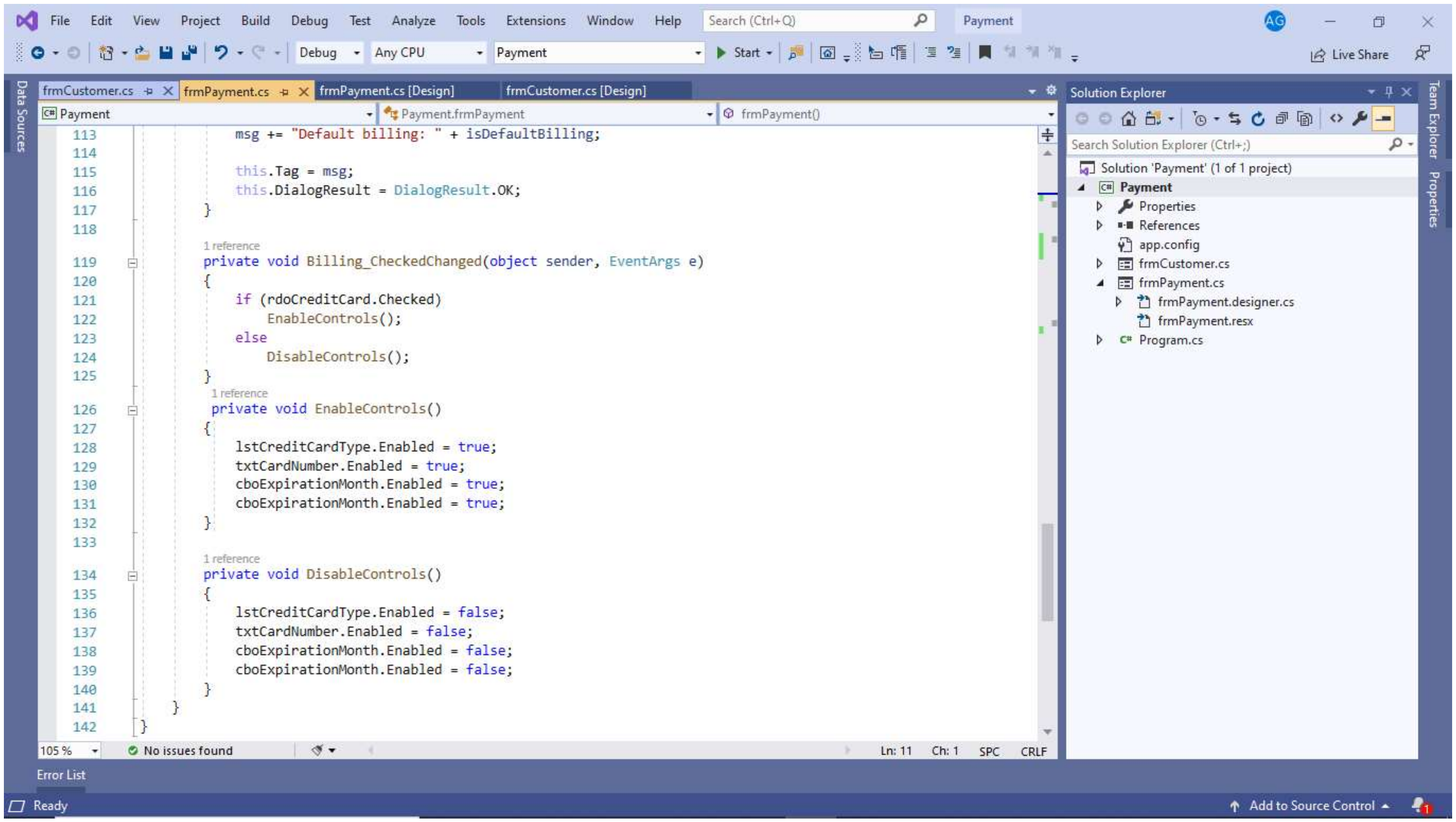
```
            int year = DateTime.Today.Year;
            int endYear = year + 8;
            cboExpirationYear.Items.Add("Select a Year...");
            while (year < endYear)
            {
                cboExpirationYear.Items.Add(year);
                year++;
            }
            cboExpirationYear.SelectedIndex = 0;
        }

        1 reference
        private void btnOk_Click(object sender, EventArgs e)
        {
            if (IsValidData())
            {
                this.SaveData();
            }
        }

        1 reference
        private bool IsValidData()
        {
            if (rdoCreditCard.Checked)
            {
                if (lstCreditCardType.SelectedIndex == -1)
                {
                    MessageBox.Show("You must enter a credit card type.",
                        "Entry Error");
                    lstCreditCardType.Focus();
                    return false;
                }
                if (txtCardNumber.Text == "")
```



```
                if (txtCardNumber.Text == "")
                {
                    MessageBox.Show("You must enter a credit card number.",
                        "Entry Error");
                    txtCardNumber.Focus();
                    return false;
                }
                if (cboExpirationMonth.SelectedIndex == 0)
                {
                    MessageBox.Show("You must select a Month.",
                        "Entry Error");
                    cboExpirationMonth.Focus();
                    return false;
                }
                if (cboExpirationYear.SelectedIndex == 0)
                {
                    MessageBox.Show("You must select a year.",
                        "Entry Error");
                    cboExpirationYear.Focus();
                    return false;
                }
            }
            return true;
        }

        1 reference
        private void SaveData()
        {
            string msg = null;
            if (rdoCreditCard.Checked == true)
            {
                msg += "Charge to credit card." + "\n\n";
                msg += "Card type: " + lstCreditCardType.Text + "\n";
```

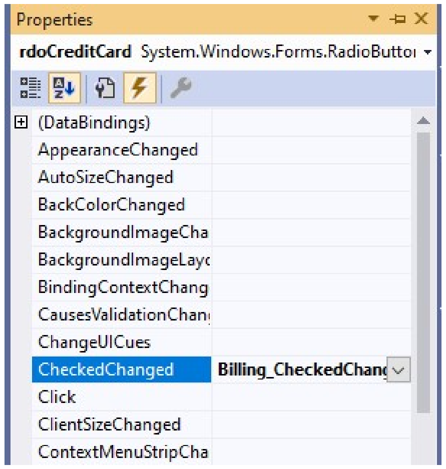


```
        string msg = null;
        if (rdoCreditCard.Checked == true)
        {
            msg += "Charge to credit card." + "\n\n";
            msg += "Card type: " + lstCreditCardType.Text + "\n";
            msg += "Card number: " + txtCardNumber.Text + "\n";
            msg += "Expiration date : "
                + cboExpirationMonth.Text + "/"
                + cboExpirationYear.Text + "\n";
        }
        else
        {
            msg += "Send bill to customer." + "\n";
        }

        bool isDefaultBilling = chkDefault.Checked;
        msg += "Default billing: " + isDefaultBilling;

        this.Tag = msg;
        this.DialogResult = DialogResult.OK;
    }

    1 reference
    private void Billing_CheckedChanged(object sender, EventArgs e)
    {
        if (rdoCreditCard.Checked)
            EnableControls();
        else
            DisableControls();
    }
    1 reference
    private void EnableControls()
    {
```



```
            msg += "Default billing: " + isDefaultBilling;

            this.Tag = msg;
            this.DialogResult = DialogResult.OK;
        }

        1 reference
        private void Billing_CheckedChanged(object sender, EventArgs e)
        {
            if (rdoCreditCard.Checked)
                EnableControls();
            else
                DisableControls();
        }
         1 reference
         private void EnableControls()
        {
            lstCreditCardType.Enabled = true;
            txtCardNumber.Enabled = true;
            cboExpirationMonth.Enabled = true;
            cboExpirationMonth.Enabled = true;
        }

        1 reference
        private void DisableControls()
        {
            lstCreditCardType.Enabled = false;
            txtCardNumber.Enabled = false;
            cboExpirationMonth.Enabled = false;
            cboExpirationMonth.Enabled = false;
        }
    }
}
```

Solution 'Payment' (1 of 1 project)
- Payment
  - Properties
  - References
  - app.config
  - frmCustomer.cs
  - frmPayment.cs
    - frmPayment.designer.cs
    - frmPayment.resx
  - Program.cs



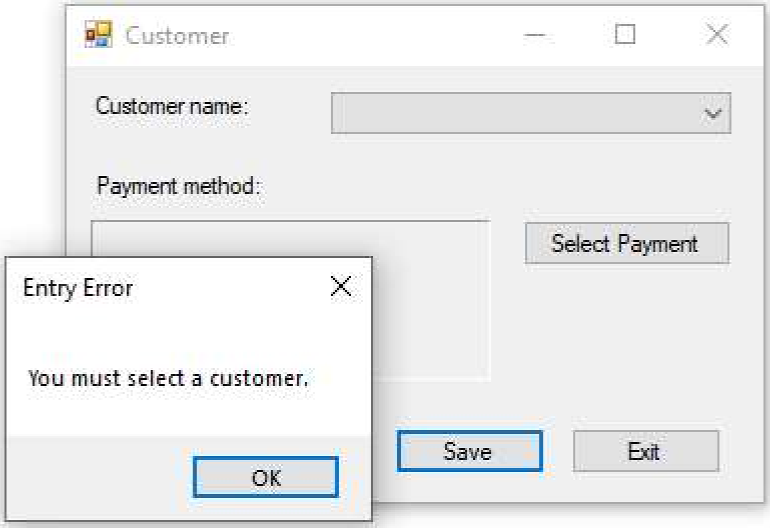
Properties

rdoCreditCard System.Windows.Forms.RadioButto

(DataBindings)	
AppearanceChanged	
AutoSizeChanged	
BackColorChanged	
BackgroundImageCha	
BackgroundImageLay	
BindingContextChang	
CausesValidationChan	
ChangeUICues	
CheckedChanged	Billing_CheckedChang
Click	
ClientSizeChanged	
ContextMenuStripCha	

Test the Application

1) No Customer Selected 2) Cust Selected, No payment selected 3) Bill to customer selected 4)No Card # entered 5) No month selected


Customer

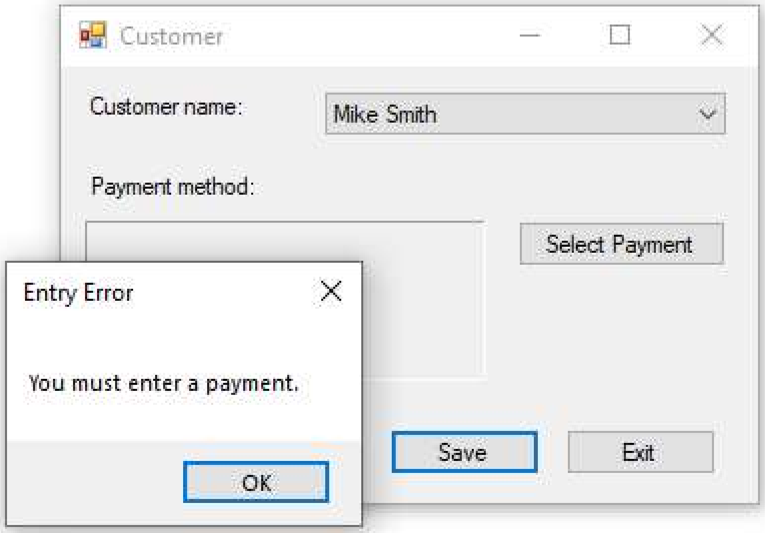
Customer name:

Payment method:

Entry Error

You must select a customer.

1.


Customer

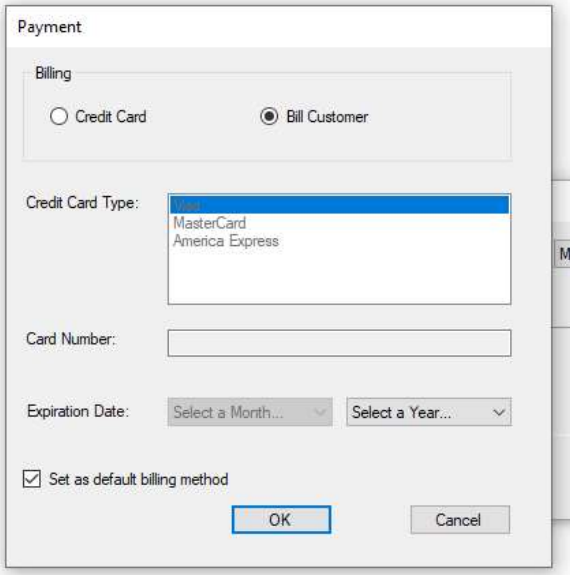
Customer name: Mike Smith

Payment method:

Entry Error

You must enter a payment.

2.


Payment

Billing

Credit Card [✓ Bill Customer]

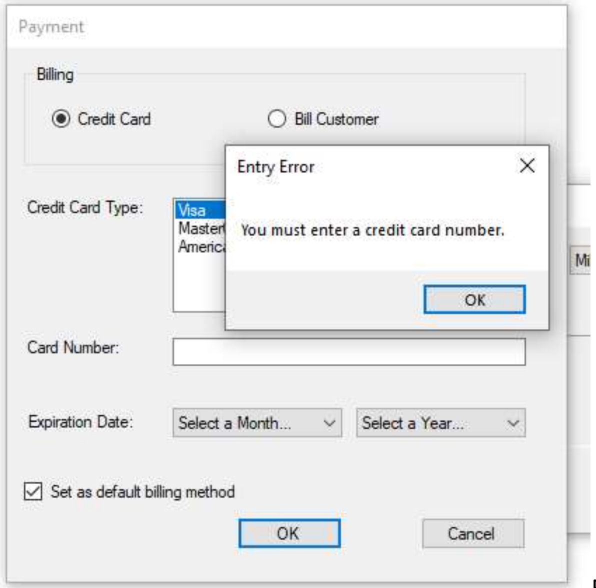
Credit Card Type: Visa, MasterCard, America Express

Card Number:

Expiration Date: Select a Month... Select a Year...

[✓] Set as default billing method

3.


Payment

Billing

[✓ Credit Card] Bill Customer

Credit Card Type: Visa, Master, Americ

Entry Error

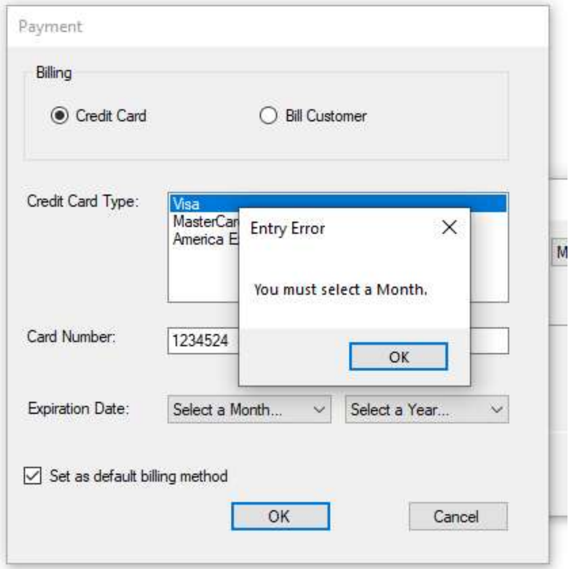
You must enter a credit card number.

Card Number:

Expiration Date: Select a Month... Select a Year...

[✓] Set as default billing method

4.


Payment

Billing

[✓ Credit Card] Bill Customer

Credit Card Type: Visa, MasterCar, America E

Entry Error

You must select a Month.

Card Number: 1234524

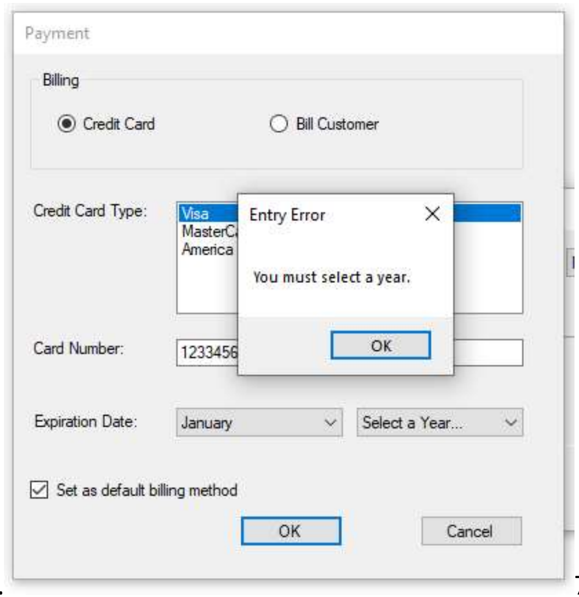
Expiration Date: Select a Month... Select a Year...

[✓] Set as default billing method

5.

Test the application Cont.

6) No Year selected 7) Completed payment form, set as default checked 8) Completed payment form, set as default unchecked

6. 

Payment

Billing

(●) Credit Card ( ) Bill Customer

Credit Card Type: Visa, MasterC, America

Entry Error

You must select a year.

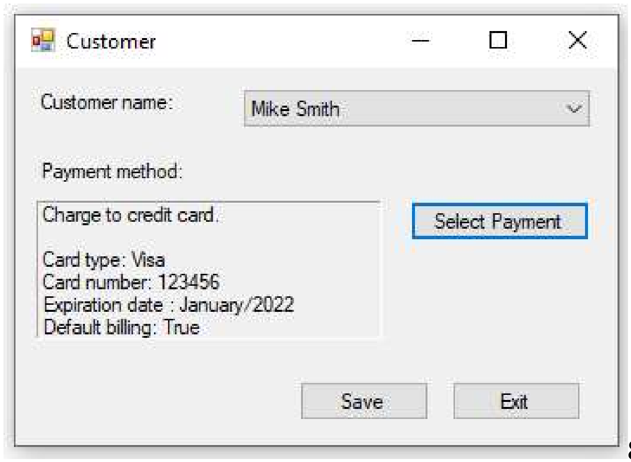
OK

Card Number: 1233456

Expiration Date: January | Select a Year...

[✓] Set as default billing method

OK Cancel

7. 

Customer

Customer name: Mike Smith

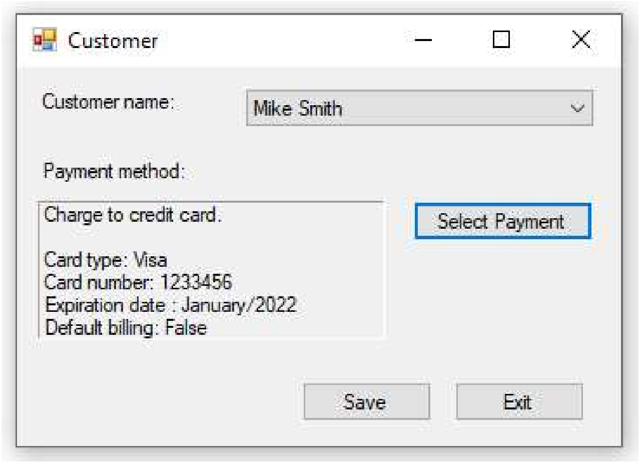
Payment method:

Charge to credit card.

Card type: Visa
Card number: 123456
Expiration date : January/2022
Default billing: True

Select Payment

Save Exit

8. 

Customer

Customer name: Mike Smith

Payment method:

Charge to credit card.

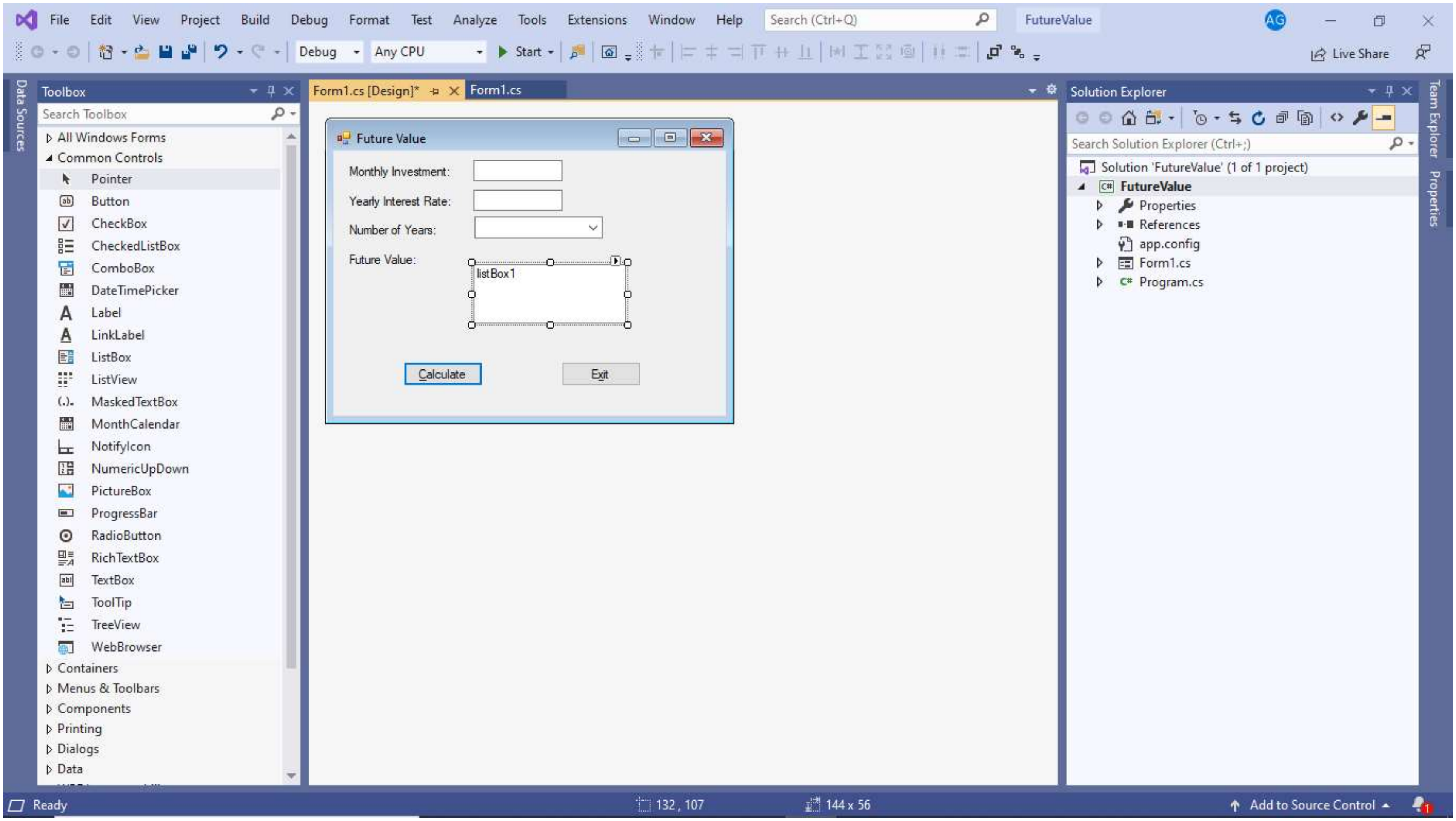
Card type: Visa
Card number: 1233456
Expiration date : January/2022
Default billing: False

Select Payment

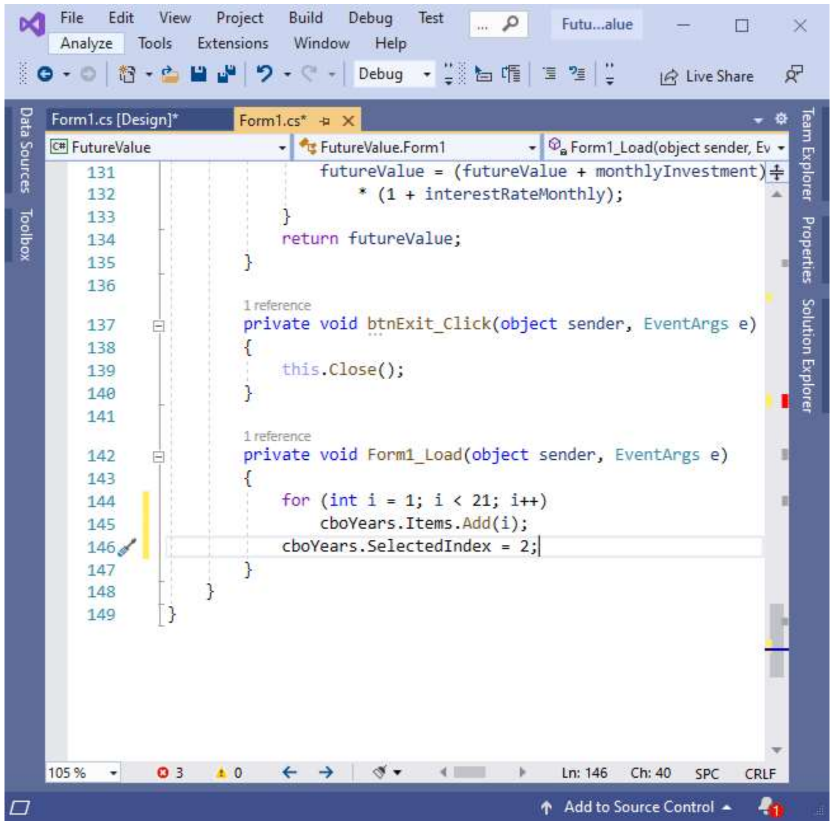
Save Exit

Exercise 10-2 Enhance the FV application

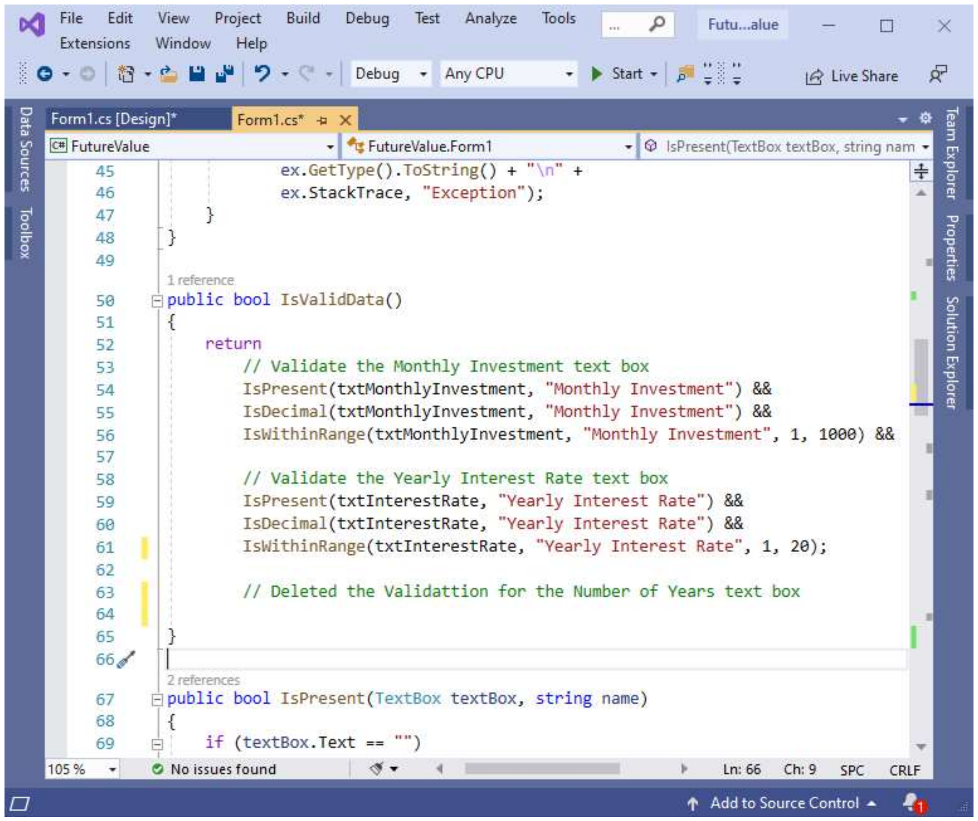
Open the future value



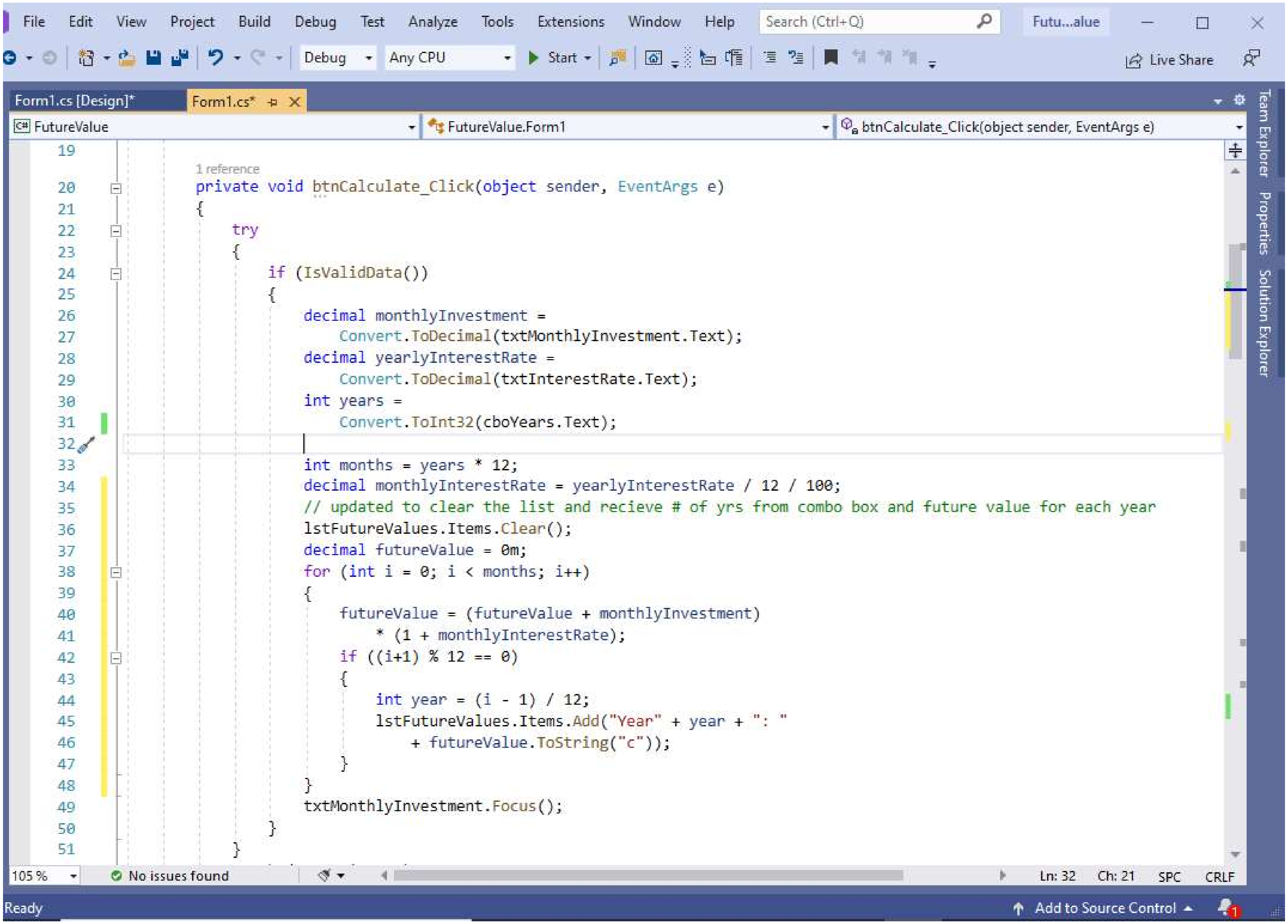
Add Code that works with the controls



```
                futureValue = (futureValue + monthlyInvestment)
                    * (1 + interestRateMonthly);
            }
            return futureValue;
        }

        1 reference
        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        1 reference
        private void Form1_Load(object sender, EventArgs e)
        {
            for (int i = 1; i < 21; i++)
                cboYears.Items.Add(i);
            cboYears.SelectedIndex = 2;
        }
    }
}
```



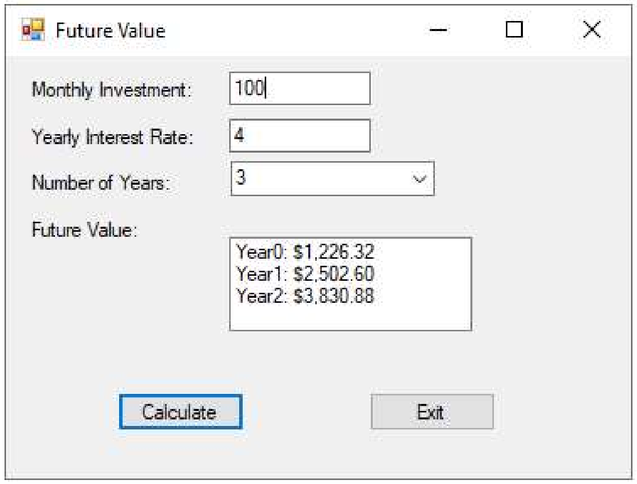
```
                    ex.GetType().ToString() + "\n" +
                    ex.StackTrace, "Exception");
            }
        }

        1 reference
        public bool IsValidData()
        {
            return
                // Validate the Monthly Investment text box
                IsPresent(txtMonthlyInvestment, "Monthly Investment") &&
                IsDecimal(txtMonthlyInvestment, "Monthly Investment") &&
                IsWithinRange(txtMonthlyInvestment, "Monthly Investment", 1, 1000) &&

                // Validate the Yearly Interest Rate text box
                IsPresent(txtInterestRate, "Yearly Interest Rate") &&
                IsDecimal(txtInterestRate, "Yearly Interest Rate") &&
                IsWithinRange(txtInterestRate, "Yearly Interest Rate", 1, 20);

                // Deleted the Validattion for the Number of Years text box

        }

        2 references
        public bool IsPresent(TextBox textBox, string name)
        {
            if (textBox.Text == "")
```



```
            1 reference
            private void btnCalculate_Click(object sender, EventArgs e)
            {
                try
                {
                    if (IsValidData())
                    {
                        decimal monthlyInvestment =
                            Convert.ToDecimal(txtMonthlyInvestment.Text);
                        decimal yearlyInterestRate =
                            Convert.ToDecimal(txtInterestRate.Text);
                        int years =
                            Convert.ToInt32(cboYears.Text);

                        int months = years * 12;
                        decimal monthlyInterestRate = yearlyInterestRate / 12 / 100;
                        // updated to clear the list and recieve # of yrs from combo box and future value for each year
                        lstFutureValues.Items.Clear();
                        decimal futureValue = 0m;
                        for (int i = 0; i < months; i++)
                        {
                            futureValue = (futureValue + monthlyInvestment)
                                * (1 + monthlyInterestRate);
                            if ((i+1) % 12 == 0)
                            {
                                int year = (i - 1) / 12;
                                lstFutureValues.Items.Add("Year" + year + ": "
                                    + futureValue.ToString("c"));
                            }
                        }
                        txtMonthlyInvestment.Focus();
                    }
                }
```

Test the application

1) Completed form correctly 2) showing 3 as default with whole list 3) no month chosen 4) interest rate invalid 5)completed form w/scroll on list box

1. 

Future Value

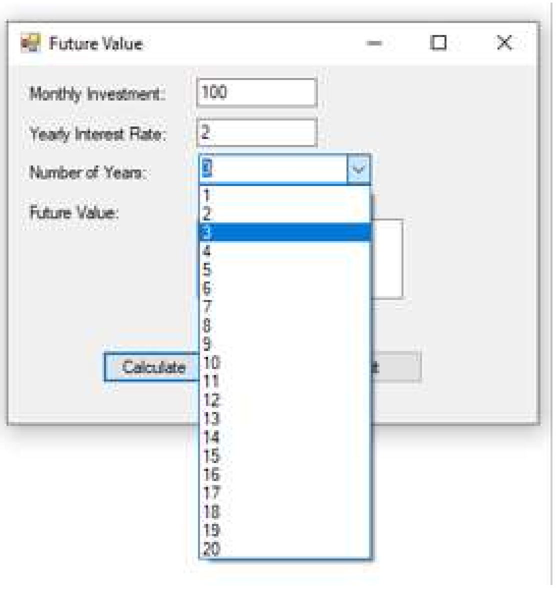
Monthly Investment: 100

Yearly Interest Rate: 4

Number of Years: 3

Future Value:

```
Year0: $1,226.32
Year1: $2,502.60
Year2: $3,830.88
```

2. 

Future Value

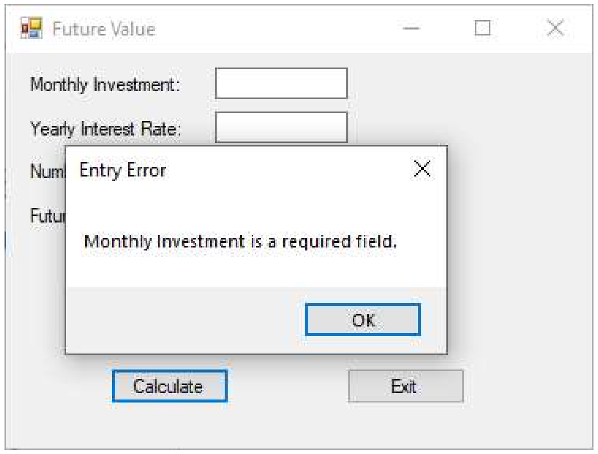
Monthly Investment: 100

Yearly Interest Rate: 2

Number of Years: 3

Future Value:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

3. 

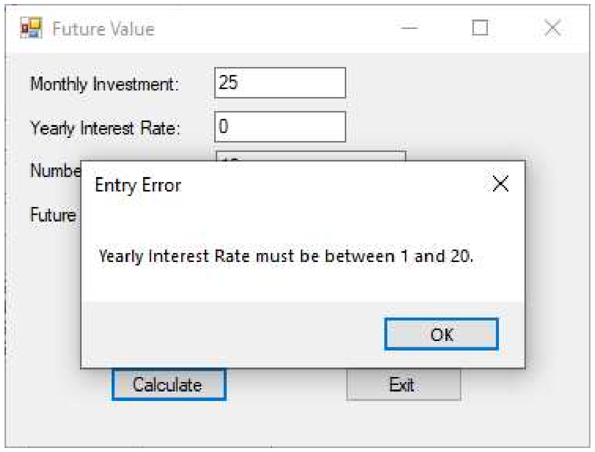
Future Value

Monthly Investment:

Yearly Interest Rate:

Entry Error

Monthly Investment is a required field.

4. 

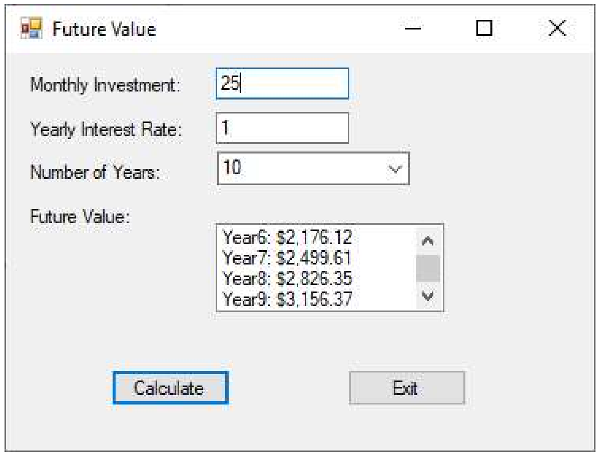
Future Value

Monthly Investment: 25

Yearly Interest Rate: 0

Entry Error

Yearly Interest Rate must be between 1 and 20.

5. 

Future Value

Monthly Investment: 25

Yearly Interest Rate: 1

Number of Years: 10

Future Value:

```
Year6: $2,176.12
Year7: $2,499.61
Year8: $2,826.35
Year9: $3,156.37
```